T.C.
SÜLEYMAN DEMİREL ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BENCHMARK ALGORİTMALARI VE BENCHMARK UYGULAMASI GELİŞTİRME

Hüseyin Bilal MACİT

Danışman: Yrd. Doç. Dr. Arif KOYUN

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
ISPARTA - 2012

İÇİNDEKİLER

ÖZET

Yüksek Lisans Tezi

BENCHMARK ALGORİTMALARI VE BENCHMARK UYGULAMASI GELİŞTİRME

Hüseyin Bilal MACİT

Süleyman Demirel Üniversitesi

Fen Bilimleri Enstitüsü

Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Yrd. Doç. Dr. Arif KOYUN

Bu çalışmanın amacı, bilgisayar bileşenlerinin performans kıyaslanmasında kullanılan algoritmaları ve geliştirilmiş yazılımları incelenmek ve yeni bir benchmark uygulaması geliştirmektir.

Bu çalışmada, günümüzde satışı yapılmakta olan benchmark yazılımları incelenmiş ve Borland Delphi yazılım geliştirme platformunda yeni bir benchmark uygulaması geliştirilmiştir. Geliştirilen uygulama, tek tuşla 22 farklı algoritma ile 43 farklı test yaparak sistemin matematiksel işlem, mantıksal işlem, grafik, bellek ve sabit disk performanslarını ölçmekte, test sonuçları ile donanım bileşenlerine sayısal skor vererek veri tabanına kaydetmektedir.

Geliştirilen yazılım ile yaklaşık yüz farklı bilgisayar üzerinde testler yapılarak bir veri tabanı oluşturulmuş ve sistemlerin başarımı birbiriyle kıyaslanmıştır.

Geliştirilen yazılımın başarımı, standartlaşmış benchmark algoritmaları ile test yapan firmaların hem skor veri tabanları ile kıyaslanmış, hem de eşdeğer sistemlerde karşılıklı uygulanarak test edilmiştir.

Anahtar Kelimeler: Benchmark, Algoritma

2012, 94 sayfa

ABSTRACT

M.Sc. Thesis

BENCHMARK ALGORITHMS AND DEVELOPING A BENCHMARK APPLICATION

Hüseyin Bilal MACİT

Süleyman Demirel University

Graduate School of Applied and Natural Sciences

Computer Engineering Department

Supervisor: Asst. Prof. Dr. Arif KOYUN

The purpose of this study is to review the algorithms used for comparison of performance of computer components and review the software designed before and develop a new benchmark application.

This study examined benchmark applications sale held today and developed a new benchmark application with Borland Delphi software developing platform. This new application measures the system' s mathematical processing, logic processing, graphics, memory and hard drive performance with 43 different tests in 22 different algorithms and it notes the hardware to database by giving numerical score.

With the software developed, about a hundred of different computer systems tested and a database generated to compare each computer.

To measure the performance of the software developed, it is compared with scores and databases of firms that are testing computers with standardized algorithms and it is tested by applying mutually equivalent systems.

Keywords: Benchmark, Algorithm

2012, 94 pages

TEŞEKKÜR

Bu araştırma için beni yönlendiren, karşılaştığım zorlukları bilgi ve tecrübesi ile aşmamda yardımcı olan değerli danışman hocam Yrd. Doç. Dr. Arif KOYUN' a teşekkürlerimi sunarım.

2916-YL-11 no' lu proje tezimi, maddi olarak destekleyen Süleyman Demirel Üniversitesi Bilimsel Araştırma Projeleri Yönetim Birimi Başkanlığı' na teşekkür ederim.

Çalışma süresince manevi desteklerini esirgemeyen eşime ve aileme teşekkürü bir borç bilirim.

Hüseyin Bilal MACİT

ISPARTA, 2012

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

SİMGELER VE KISALTMALAR

SİMGELER

∑	Toplam
π	Pi
σ	Sigma

KISALTMALAR

RISC	Reduced Instruction Set Computer
VLIW	Very Long Instruction Word
AMD	Advanced Micro Devices
SI	Norton SysInfo
TPC	Transaction Processing Performance Council
TCO	Total Cost of Ownership
CPU	Central Processing Unit
SPEC	Standard Performance Evaluation Corporation
EEMBC	Embedded Microprocessor Benchmark Consortium
HTTP	Hyper Text Transfer Protocol
I / O	Input Output
HDD	Hard Disk Drive
BDE	Borland Database Engine
PSP	Paint Shop Pro
GPU	Graphics Processing Unit
WMI	Windows Management Instrumentation
OPENGL	Open Graphics Library
BMP	Bitmap
PCI	Peripheral Component Interconnect
3D	3 Dimension
RAM	Random Access Memory

1. GİRİŞ

20. yüzyılın ortalarında bir odaya sığmayacak kadar büyük olan bilgisayarlar, çalışmak için binlerce watt gücünde elektrik kaynağına ihtiyaç duymakta, ancak sadece toplama çıkarma gibi basit matematiksel işlemleri yapabilmektedir. Delikli kartlar, boşluk tüpleri gibi basit teknolojilerden sonra transistörlerin kullanılmasıyla bilgisayarların boyutu küçülmeye başlamış, üretim maliyetleri, enerji sarfiyatları azalmış, işlem kapasiteleri artmıştır. 21. Yüzyıl başlarında gelişmiş ülkelerde neredeyse her eve girmiş olan bilgisayarlar, kurumsal alanda da ciddi boyutta kullanılmaya başlamıştır. Günümüzde internetin de yaygınlaşması ile neredeyse kişi başına bir bilgisayar düşmektedir. Bununla beraber günümüzde bilgisayarlar; masaüstü, dizüstü ve tablet gibi, ihtiyaca göre farklı formlarda satılmaktadır.

Bilgisayar ve bileşenlerinin üretiminde marka çeşitliliğinin artması rekabeti doğurmuş, rekabet de ürün çeşitliliğinin artmasını sağlamıştır. Tüketicinin sorunu; bilgisayarı veya bileşenlerini satın alırken hangi kriterlere göre ürün satın alacağını seçmek, üreticinin sorunu da ürününü nasıl pazarlayacağını ve bir sonraki ürünü bir öncekinden nasıl daha hızlı hale getireceğini bulmak olmuştur. Bu sorunun çözümünde hem tüketiciye hem üreticiye yardımcı olacak en iyi çözüm benchmark yazılımları olmuştur. Tüketici ürün satın alırken, satın almak istediği ürünün benchmark sonuçlarına göre karar vermekte, üretici ürün geliştirirken ve pazarlarken yine benchmark sonuçlarını baz almaktadır.

Bir bilgisayarın veya uygulamanın ne kadar başarılı olduğu, üç kritere göre nitelendirilir (Okulicka, 2010);

- Hız
- Fiyat
- Verimlilik

Bu çalışmada, bir benchmark yazılımının nasıl oluşturulduğunu incelemek ve örnek bir benchmark yazılımı geliştirmek amaçlanmıştır. Benchmark yazılımı ile bilgisayar

bileşenlerine belli bir işlem yükü yüklenmekte, bileşenin bu yükü ne kadar zamanda çözdüğü hesaplanmaktadır. Geliştirilen uygulamada, anlaşılırlık, kullanım kolaylığı, çok sayıda algoritmayı az zamanda gerçekleştirmek ve dünya çapında kabul görmüş benchmark yazılımları ile karşılaştırıldığında benzer sonuçlar alınabilmesi hedeflenmiştir.

1.1. Benchmarking Kavramı

Benchmarking sözcüğü, "benchmark" tan gelmektedir. Benchmark bir ölçü birimi ve benchmarking ise bu ölçümün yapılmasıdır. Henüz Türkçe' de karşılığı tam olarak benimsenmemiş olsa da, benchmarking; referans alma, örnek alma ve en çok da "kıyaslama" olarak kullanılmaktadır (Başlıgil, 2009).

Benchmark' ın gerçek dünyada ilkel kullanım alanındaki anlamı ise, arazi üzerinde araştırma yapanların bir kaya, duvar yada bina üzerinde yaptıkları bir nirengi işaretine verilen addır. Arazi üzerinde ölçüm yapanlar, daha sonra bu işareti, diğer ölçümleri yapabilmek için referans noktası olarak kullanmaktadırlar. Bu referans noktasına bench-mark adını vermekte, bundan sonraki işaretlerin referans noktasına uzaklığına göre hesaplamalar yapmaktadırlar.

Benchmarking kelimesinin literatürde teknik bir anlam kazanması, 1970'li yıllarda, iş dünyasında kullanılmaya başlamasıyla olmuştur ve teknik terim olarak ilk önce işletme literatürüne girmiştir. İşletme biliminde benchmark' ın tanımı; seçilen konuda "en iyi" olabilmek için kendi ürünlerini, hizmetlerini ve uygulamalarını, rakipleri ya da endüstrinin önde gelen diğer şirketleri ile beraber değerlendirerek gelişme sürecidir (Baş, 2012).

Endüstride, iki firma arasında kıyaslama uygulaması yapabilmek için, iki şirketin işlem ve yöntemlerine ilişkin bilgileri paylaşma konusunda önceden anlaşmış olmaları gerekir. Her iki şirket de bilgi değişiminden bazı kazançlar beklediği için dürüst sonuçlar vermek durumundadır.

Bilgisayar dünyasında benchmark, bir bileşenin başarımını, o bileşen üzerinde çeşitli sınamalar yaparak ölçmek veya tüm sistemin başarımını, farklı algoritmalar kullanarak ölçmek ve başka bileşenlerle kıyaslanabilecek şekilde skorlar üretmektir.

Tüketici, bir bilgisayar bileşenini satın alırken farklı marka ve modeller arasında tercih yapmak durumundadır. Bu tercihi yaparken kullanılan kritere fiyat/performans denir. Fiyat her zaman belli olduğuna göre, önemli olan bileşenin performansını ölçebilmektir. Bunun için firmaların kendi verdiği performans değerlerine güvenmek yerine tarafsız test yazılımlarının sonuçlarına göre hareket etmek daha doğru olmaktadır (Kapoor, 2012).

1.2. Benchmarking' in Amacı

Bilgisayar mimarisinde, bilgisayarların başarımlarını sadece özelliklerine bakarak belirlemek imkânsızdır. Bu nedenle, farklı testler geliştirilmiştir. Bu testler farklı sistemlerde çalıştırılarak, elde edilen sonuçlar farklı mimarilerin karşılaştırılmasında kullanılmaktadır.

Benchmark testlerinde, çoğunlukla yaygın kullanılan iki marka işlemcinin (AMD ve Intel) de benzer başarım gösterdiği görülmektedir. Benchmarklar, belirli bir sisteme ya da bileşene, belli bir tür iş yükü yükleyecek şekilde tasarlanırlar. Yapay (Synthetic) benchmarklar bir bileşenin üzerine bu iş yükünü yüklemek için yazılan özel programlardır. Uygulama (Application) tür benchmarklar sistem üzerinde gerçek programlar çalıştırırlar. Uygulama türündeki benchmarklar bir sistemin başarımı ölçümünde etkiliyken, Yapay benchmarklar ise sistem bileşenlerinin (sabit disk, ağ aygıtları, I/O aygıtları gibi) tek başlarına başarımlarının ölçümünde etkilidirler.

Benchmarklar özellikle yarı iletken mikroişlemcilerin tasarımında önemlidir. Benchmarklar sayesinde tasarımcılar, mikro mimari düzeyinde aldıkları kararları ölçüp dengelerler. Bilgisayar tasarımcılarının tasarladıkları sistemlerin başarımlarını ölçmek için benchmarkları kullanması oldukça uzun bir geçmişe sahiptir. Ancak

önceleri bu benchmark testlerinin sonuçları gerçek kullanımla elde edilecek başarımdan farklı olmuştur. 1980'lerde bazı derleyiciler bilinen kayan nokta işlemlerinde kullanılan özel bir matematik işlemini belirleyip daha hızlı olan matematiksel eşitlik işlemleriyle değiştirmişlerdir. Bu değişim 1990' lı yılların ortasına kadar benchmarklar dışında kullanışlı olmuştur. Ancak bu kullanışlılık RISC ve VLIW mimarilerinin tasarımcıları derleyici teknolojisinin başarımla ilgisini fark etmelerine kadar sürmüştür. Günümüzde benchmarklar; derleyici şirketleri tarafından sadece kendi benchmark sonuçlarını yükseltmek için değil, aynı zamanda tasarladıkları uygulamaların başarımlarını artırmak için de kullanılmaktadır.

Benchmarking' in amaçları kısaca aşağıdaki gibidir;

- Fiyat / performans değerini belirlemek
- Tüketicinin satın alma maliyetini düşürmek
- Uygulamanın çalışabileceği en iyi sistemi saptamak
- Mevcut sistemden en yüksek verimi almak
- Üreticiler arasındaki rekabet avantajını kullanmak
- Üreticinin yeni ürünlerini tarafsız tanıtma platformunu oluşturmak

1.3. Bench-marketing

Üreticiler çoğunlukla sadece bazı benchmarkları ya da benchmarkların belirli yönlerini kullanarak ürünlerini en iyi şekilde tanıtmaya çalışmaktadırlar. Ayrıca bazıları bu benchmarkların sonuçlarını veya özelliklerini değiştirerek ürünlerini olabildiğince iyi şekilde göstermeye çalışmaktadırlar. Bazı firmalar kendi ürünlerinin skorlarını yüksek çıkarabilecek özel benchmark yazılımları üretmektedir. Bu eylemlerin tamamına İngilizce bench-marketing adı verilmektedir (Kapoor, 2012).

1.4. Benchmarking Kavramının Geçmişi ve Ortaya Çıkışı

Benchmarking kelimesi, ilk olarak Xerox şirketi tarafından kullanılmaya başlanmıştır (Çizelge 1.1). 1956′da ilk fotokopi makinesini pazara sunan Xerox, o dönemde

piyasada rekabet edebilecek güçlü rakiplerinin olmayışından dolayı, müşteri isteklerini çok fazla ön planda tutmamıştır. Fakat 1970′lerin ortasında pazara kullanımı kolay ve daha ucuz ürünlerle giren Japon üreticileri Xerox'u rahatsız etmiştir. Çünkü 1970′den 1980′ e Xerox'un gelirleri %95′ ten %46′ ya, pazar payları da %49′ dan %22′ ye düşmüştür. Bunun üzerine araştırmalar başlamıştır. Japon ürünlerinin araştırılması sonucu görmüşlerdir ki Japonlar daha hızlı, maliyeti daha düşük ve daha kaliteli ürünleri pazara sunmuşlardır. Japonlar tabi ki Xerox'a fabrikalarını açmamışlardır, ancak Xerox Japon ürünlerini incelemek için, Japon ürünlerini satın alıp parçalayarak, nasıl üretildiklerini incelemiştir. Rakiplerinin iş süreçlerini bu sayede daha iyi görmüş, kendi üretimleriyle kıyaslama şansı bulmuş ve onları örnek almaya yani benchmarking' i uygulamaya başlamıştır.

Çizelge 1.1. Xerox'un uyguladığı benchmarking örnekleri.

Xerox'un benchmarking ortakları	Benchmarking yapılan süreç
American Express	Faturalama ve Tahsilat
American Hospital Supply	Envanter Kontrolü
Florida Light and Power	Kalite Güvencesi Süresi
Ford Motor Company	Üretim Hattı Dizaynı
General Electric	Robot Sistemi
Cummins Engine Company	Günlük Üretim Planlaması
Westinghouse	Depo kontrolü, Barkod uygulaması

(Kaynak: Yüksel, 2003)

Xerox'tan sonra, kıyaslamada kullanarak firma başarımlarını arttıran birçok işadamı vardır. Bunlara birkaç örnek verecek olursak; Ford'un kurucusu Henry Ford, yürüyen bant sistemiyle üretimi, 1912'de Chicago'da bir tanıdığını görmek için gittiği mezbahadan esinlenerek geliştirmiştir. Kasapların her birinin karkasın belirli bir bölümünü keserek, kalanını diğer arkadaşlarına devrettiğini gören Ford, aynı yöntemi otomobil yapımında da ufak bir farkla uygulamıştır. Çengellerin üzerinde kaydığı çelik ray yerine, hareketli bir band uygulaması geliştirmiştir. Yine 1950'li yıllarda Toyota'nın kurucusu, oğlu Eiji Toyoda' yı General Motors, Ford gibi otomobil

devlerini incelemek üzere A.B.D.' ye göndermiştir. Eiji Toyoda sadece bu işletmeleri ziyaret etmekle kalmamış, süpermarketlerde bile gözlemlerde bulunmuştur. Süpermarketlerde boşalan rafların geceleri hızla ve ihtiyaç doğrultusunda doldurulmasından etkilenen Toyoda, Japonya' ya dönüşte "just-in-time" ya da tam zamanında üretim, sıfır stokla çalışma olarak adlandırılan sisteminin ilk uygulamalarını başlatmıştır.

1980'lerden sonra başta ABD olmak üzere bütün batıda yaygınlık kazanan benchmarking yöntemi, bugün de dev işletmeler tarafından kullanılmaktadır. Bunların başında ise Dell Computer ve General Electric (GE) gibi işletmeler gelmektedir. Dell Computer'ın kurucusu Michael Dell, işletmede ortaya çıkan sorunları aşmak ve geleceğe hazırlık yapmak için, değişik sektörlerden iyi uygulamaları incelediklerini söylemektedir. Ülkemizde küreselleşme ile birlikte uluslararası pazarlara girmek, rekabet gücünü artırmak, kaliteli mal ve hizmet üretmek amacıyla benchmarking yeni bir yönetim tekniği olarak kullanılmaya başlanmıştır. Sabancı Holding bünyesindeki işletmeler; Beksa, Brisa, Dusa, Kordsa ve Olmuksa tarafından benchmarking çalışmalarına sistematik bir yapı kazandırmak amacıyla 1997'de kurulan "BENCHSA" adlı bir çalışma grubu, benchmarking' i bir süreç olarak ele almakta, sürekli iyileştirmenin üzerinde önemle durmaktadır (Baş, 2012).

1.5. Benchmarking' de Karşılaşılan Zorluklar

Benchmark yazılımları kullanılarak yapılan kıyaslamalar kolay değildir. Beklenen ve kullanışlı sonuçlar elde edebilmek için tekrarlanan döngüler içerirler. Ayrıca benchmarkların yorumlanması da oldukça zordur. Aşağıda genel olarak karşılaşılan bir takım zorluklar listelenmiştir:

- Sağlayıcılar tarafından, ürünlerini endüstriyel standartlı benchmarklara göre ayarlamaları gerekmektedir. Norton SysInfo (SI) ile esas olarak çoklu işlemlerin hızlarına yöneliminden dolayı bu ayarlama biraz daha kolay yapılmaktadır. Bu tür sonuçların yorumlanmasında dikkat edilmesi gerekmektedir.

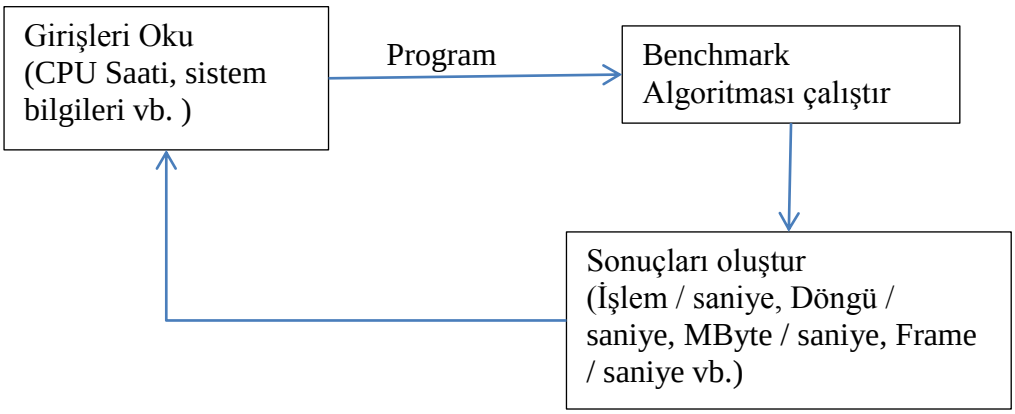
- Benchmarklar genellikle bileşenlerinin özelliklerinin başarısından başka şeylere önem vermezler. Bu özelliklere örnek olarak güvenlik, kullanılabilirlik, güvenilirlik, uygulamaların çalışmasında bütünlük, yararlılık, ölçeklenebilirlik gibi özellikler gösterilebilir. Bu özelliklerin hepsi önemlidir ve bunlar arasında denge kurulmaya çalışılmaktadır. TPC (Transaction Processing Performance Council) benchmark şartnamesine göre bu işlem ACID özellik testi, veri tabanı ölçeklenirlik kuralları ve servis seviye gereksinimi gibi işlemlerin yapılmasıyla belirlenir.
- Genellikle benchmarklar, TCO' yu (toplam sahiplik masrafı) ölçmezler. TPC benchmark şartnamesine göre TCO' nun basit şekilde formüle edilmesi için başarım ölçüsünün belirlenmesi gerekir. Bunun için fiyat / performans ölçüsünün belirlenmesi gerekir.
- Benchmarklar, karışık iş yüklerinin başarımını çoklu işlemleri tam, iş bağlamlı çoklu kısım/çoklu uygulama prensibiyle çalıştırarak ölçer. Elde edilen sonuçlar nadiren gerçek dünyadaki başarımlarına eşdeğerdir.
- Üreticiler, benchmarkların sonucunda bulunan geliştirme, test, ya da yıkım onarımı, işlem kapasitesi gibi gereksinimlerle ilgilenmezler. Sağlayıcılar özellikle ilk kazanım fiyatını olabildiğince az göstermek için gerekli en düşük ürün kapasitesini belirtmek isterler.
- Benchmarklar, üretilmiş farklı sunucular ve aşırı hassas ağ topolojilerine uyum sağlamakta sorun yaşamaktadırlar. Izgara hesaplamanın (grid computing) ortaya çıkması benchmark sınamasının yapılmasını kısmen zorlaştırmıştır.
- Kullanıcıların, benchmarkların önerdiği başarımlarda farklı şeyler anlaması mümkündür. Benchmarklar, ufak sapmalardan çok ortalama başarımlara önem vermektedir. Yani kullanıcı bakış açısından çok, bilimsel bakış açısı göz önüne alınmaktadır.

Benchmark kurumları çoğunlukla basit bilimsel metotları izlemezler. Bu da beraberinde küçük örnekleme boyutu, değişken kontrolünde zorlaşma, sonuçların tekrarlanmasında kısıtlama gibi sonuçların ortaya çıkmasına sebep olmaktadır (Kapoor, 2012).

1.6. Benchmark Yazılımları

Benchmark yazılımları; bilgisayar mimarisinde bir bileşenin başarımını, o bileşen üzerinde çeşitli sınamalar yaparak ölçmek için çalıştırılan bir veya bir grup bilgisayar programıdır. Benchmark yazılımları; bilgisayarın bir donanımının karakteristik bir özelliğinin başarımının belirlenmesinde de kullanılmaktadır (Şekil 1.1.). Örneğin Merkezi İşlem Biriminin (CPU) kayan nokta (floating point) işlemlerindeki başarımını ölçmek için kullanılmaktadır. Ayrıca bazı durumlarda bu sınama; yazılımların başarımı için de kullanılmaktadır. Yazılımlar için kullanılan benchmarklara örnek olarak derleyiciler ya da veri tabanı yönetim sistemlerinin başarımlarını ölçen programlar verilebilir.

Benchmark yazılımları, farklı yonga veya sistem mimarilerinde çalışan alt sistemlerin başarımını karşılaştırmak için yöntemler geliştirmişlerdir. Benchmark yazılımları, veri tabanı yönetim programlarının değişken durumlardaki tepkilerinin anlaşılmasında yardımcı olurlar. Farklı durumlar için farklı senaryolar üretilerek de başarım ölçülebilmektedir. Örneğin kilitlenme işlemleri, yardımcı programların başarımı, farklı veri yükleme yöntemleri, eklenen yeni kullanıcılarla program hızının değişiminin karakteristiğinin belirlenmesi ve programın yeni sürümünün çıkmasıyla gelen uygulamaların başarımının belirlenmesi için farklı senaryolar üretilerek benchmarkların farklı özellikleri ya da farklı benchmarklar kullanılabilmektedir (Kapoor, 2012).



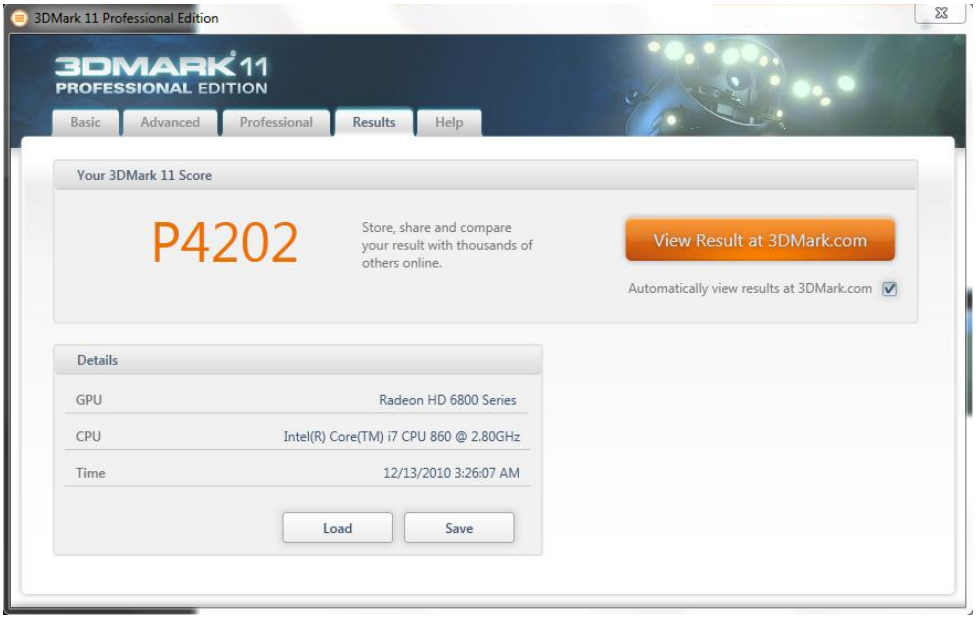
Şekil 1.1. Benchmark yazılımlarının temel yapısı (Middela, 2008).

1.7. Yaygın Kullanılan Benchmark Yazılımları

1.7.1. 3DMark / PCMark

Futuremark, çok geniş bir yelpazede benchmark sunan ve dünyanın en iyileri arasında kabul edilen bir yazılım şirketidir. İnternette bir masaüstü yada dizüstü bilgisayarın benchmark test sonuçlarını araştırıldığında, çoğunlukla ilk önce 3DMark yada PCMark skorlarına rastlamak mümkündür.

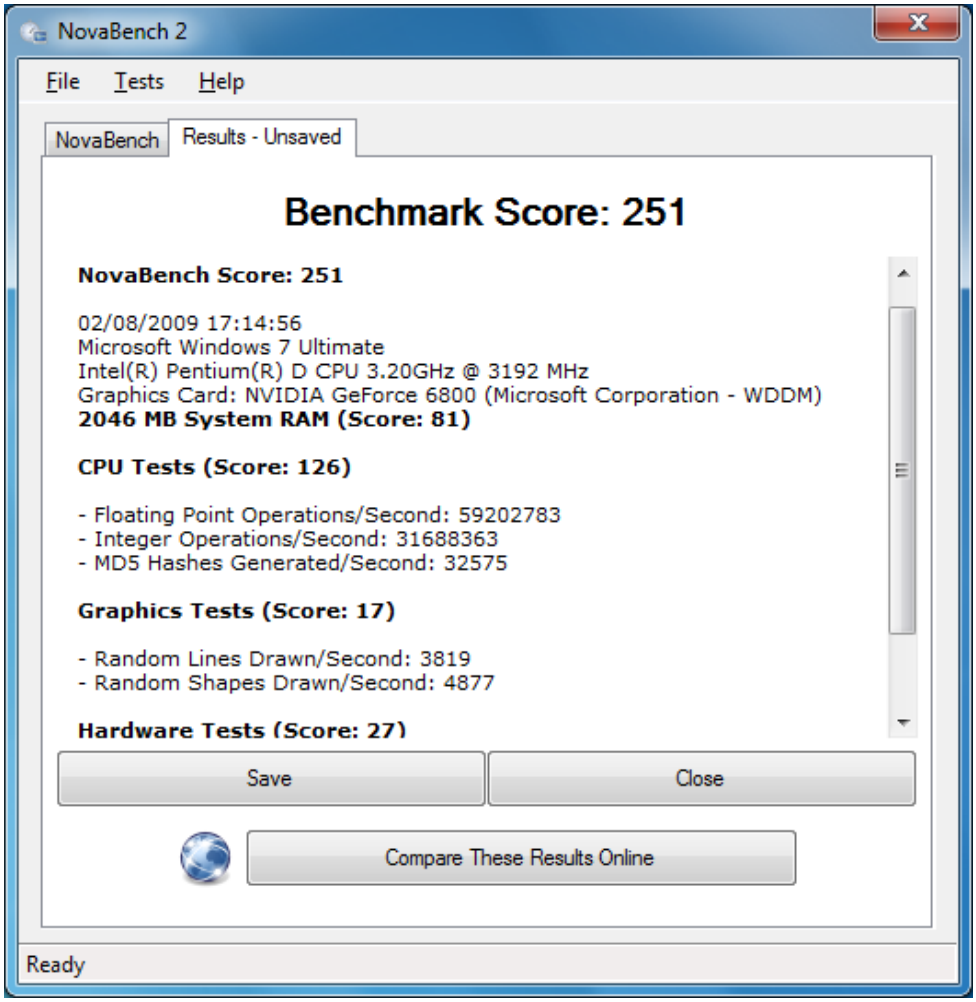
Şekil 1.2.' de 3D Mark' ın örnek ekran görüntüsü verilmiştir. 3DMark ve PCMark' ın son sürümleri ücretsiz değildir. Ancak sınırlı seçenekleri ile eski sürümleri hala ücretsiz olarak kullanılabilmektedir. 3DMark, daha çok oyun kullanıcılarına yönelik benchmark testleri içerir, PCMark ise genel bilgisayar kullanıcısına hitap eden algoritmalar içerir. Futuremark' ın puanlama algoritmaları biraz farklıdır. Çok yeni bilgisayarlar bile futuremark testlerinden düşük puanlar alabilmektedir.



Şekil 1.2. 3DMark programı ekran görüntüsü.

1.7.2. Novabench

Novabench; tamamen ücretsiz bir benchmark yazılımıdır. Novabench; işlemci hızı, 2D grafik performansı ve harddisk okuma yazma hızını test edebilmektedir (Şekil 1.3.). Her ne kadar basit ev bilgisayarları için dizayn edilmiş bir yazılım olsa da, az sayıda multimedya ve grafik testini de içermektedir.



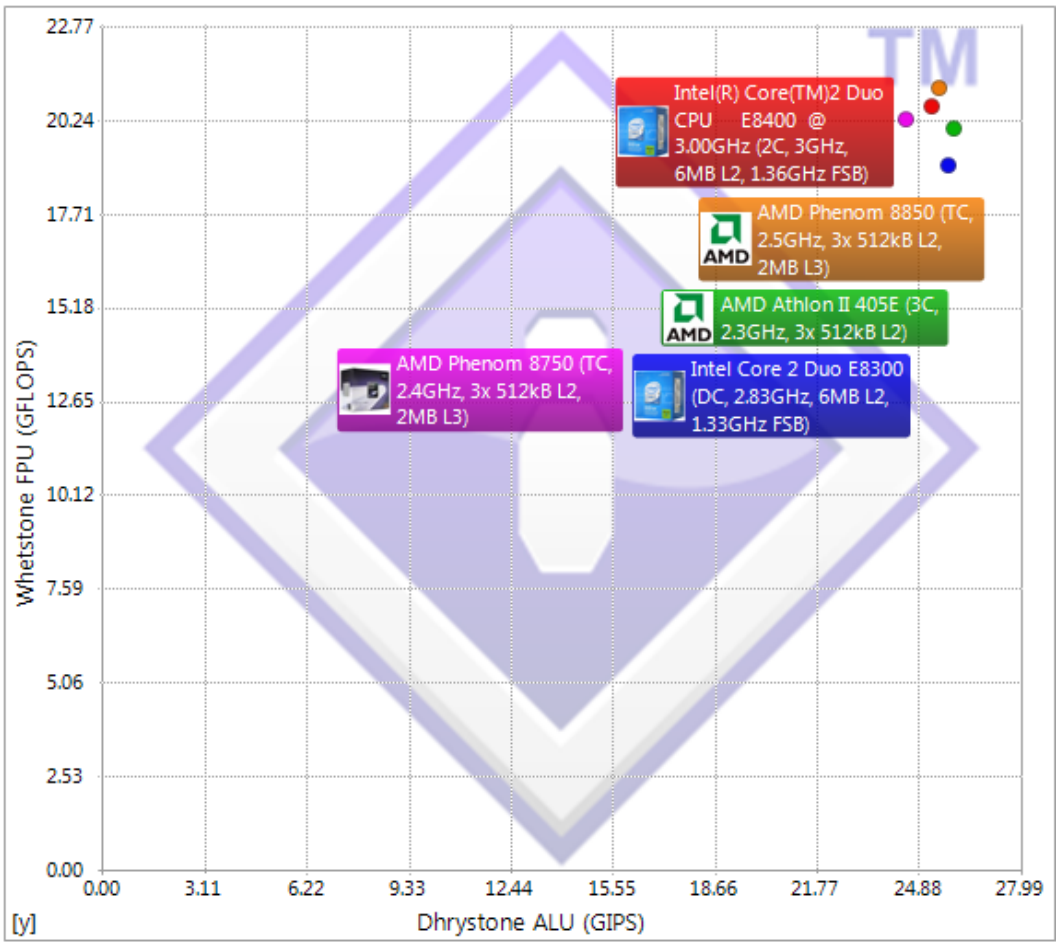
Şekil 1.3. Novabench programı ekran görüntüsü.

1.7.3. SiSoft Sandra

Sandra System Analyzer; bir bilgisayarın işleyişi hakkında ve birden çok bilgisayar üzerinde ayrıntılı bir analiz yapabilen tam donanımlı bir kriter paketidir. SiSoft Yazılım, Sandra' nın ücretsiz sürümünü herkesin kullanımına sunmuştur. Şekil 1.4.' te SiSoft Sandra' nın test sonucunda verdiği örnek ekran görüntüsü gösterilmiştir.

SiSoft Sandra; diğer bilinen benchmark uygulamalarından farklı olarak, bilgisayarın bellek bant genişliğini, ağ performansını, bilgisayarınızın güç verimliliğini gibi farklı testleri yapılabilmektedir.

Sisoft Sandra' yı diğer yazılımlardan ayıran en önemli özelliği, referansa göre sanal benchmark yapabilmesidir. Örneğin, sistemi test ettikten sonra, mevcut sistemde bir üst model veya alt model işlemciler olsa, performansın ne derece değişeceğini de göstererek, kullanıcıya upgrade durumundaki sonuçları da gösterebilmektedir. Bu özellik, sadece SiSoft Sandra' da mevcuttur (Smith, 2010).

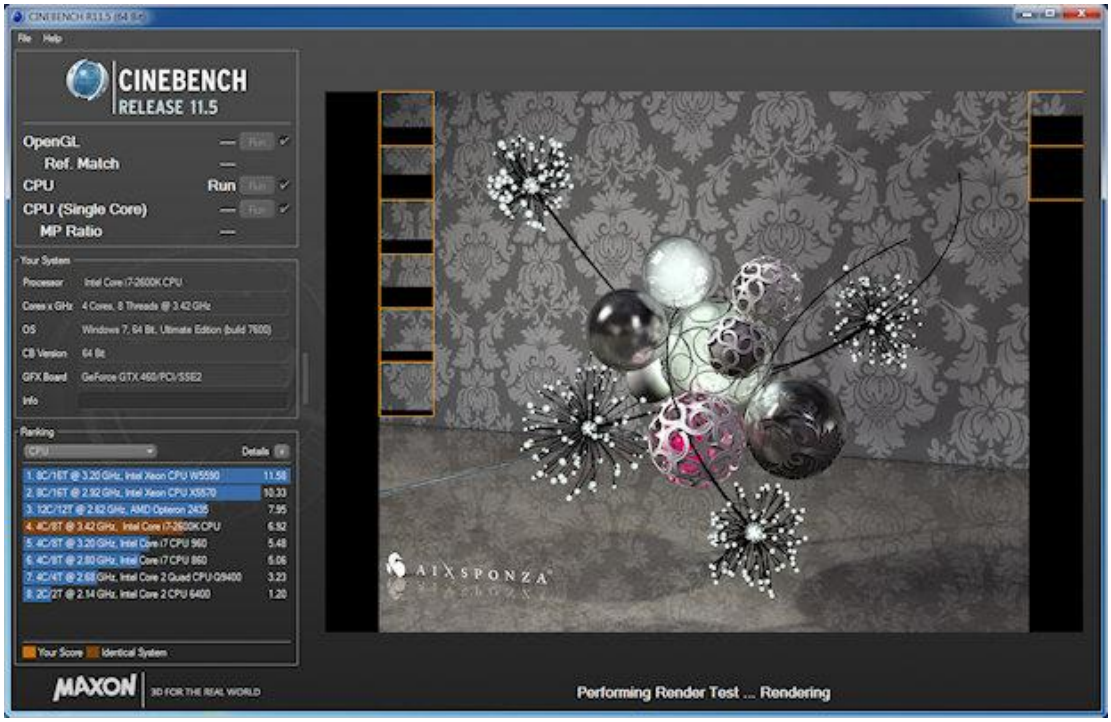


Şekil 1.4. SiSoft Sandra programının ekran görüntüsü.

1.7.4. Cinebench

3D modellemede performans, yapılan projenin istenen zamanda bitip bitmemesi ile ölçülmektedir. Cinebech'in özelliği, floating point algoritmaları kullanarak 3D Render testi yapabilmesidir. Bu işlemi yapabilen Autodesk 3DMax, Lightwave gibi pahalı programlar yerine bu ucuz programla bu test yapılabilmektedir. Cinebech,

Cinema4D adlı modelleme programının render motoru üzerine kuruludur. Cinebench, 64 ve 32 bit işletim sistemi üzerinde çalışabilmekte ve tekli veya çoklu test algoritması çalıştırma seçeneklerine sahiptir. Single-threaded ve mutli-threaded seçenekleri ile test başlatılabilir. Sonuçlar sayısal puan olarak kullanıcıya sunulur (Şekil 1.5.).



Şekil 1.5. Cinebench programının ekran görüntüsü.

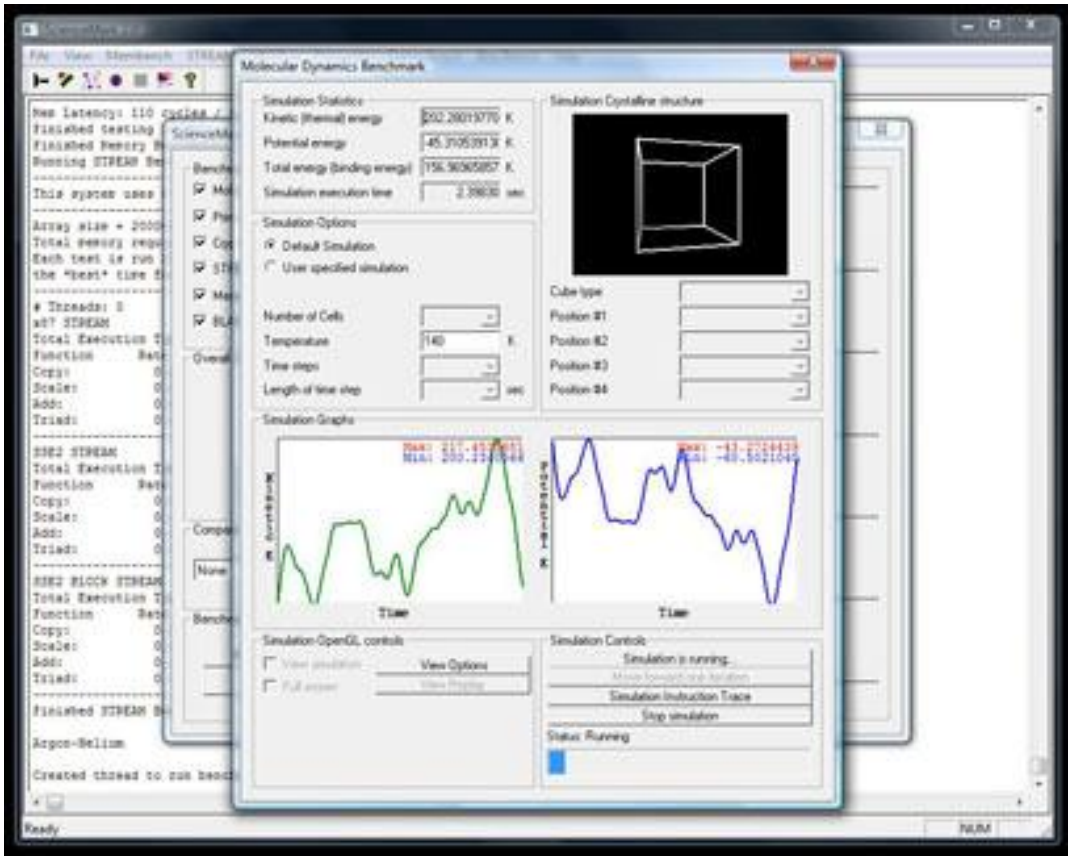
1.7.5. Fritz Chess Benchmark

Gerçek dünya yazılımına örnek olarak verilebilecek bir uygulamadır. Programın çalıştırılıp, sadece start tuşuna basılması yeterlidir. Klasik Chassbase motoru üzerine kuruludur. Test sonucunu "saniyede kaç kilo düğüm" (kilo nodes per second) olarak verir. Satranç taki olasılıklar ve hesaplar üzerine kurulu bir yazılımla bilgisayarı test etmektedir. Şekil 1.6.' da uygulamanın test sonucunda verdiği skor gösterilmektedir.

Şekil 1.6. Fritz Chess programının ekran görüntüsü.

1.7.6. ScienceMark

Sentetik benchmarklara örnek bir yazılımdır. Yaygın kullanılan mühendislik matematiği algoritmalarını kullanmaktadır. Bellek performansını, gecikme (delay) ile ölçmektedir. Kullandığı önemli testler; kriptografi, bellek gecikmesi ve moleküler dinamik' tir. Şekil 1.7.' de uygulamanın sonuç ekranı gösterilmektedir.



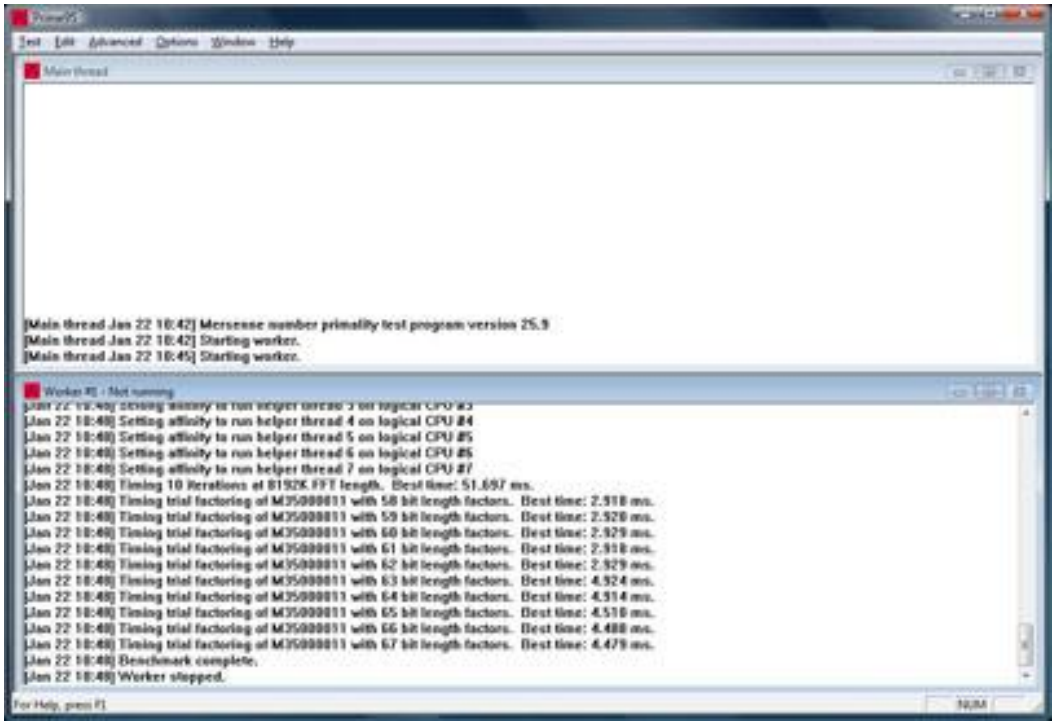
Şekil 1.7. Science Mark programının ekran görüntüsü.

1.7.7. Prime95 ve SuperPi

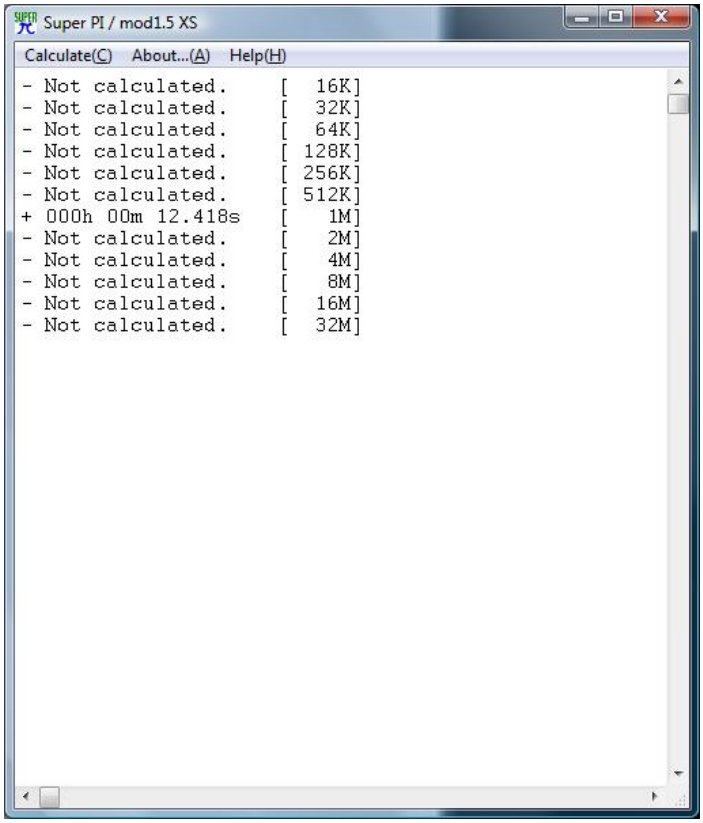
Prime95 ve SuperPi; single-threaded olmalarına rağmen, CPU matematik başarımını hesaplamak için mükemmel iki yazılımdır. Her ikisi de aslında uzun stres testleri yaptığı için, overclock tutkunları tarafından tercih edilmektedir. Her ikisi de bir performans göstergesi olarak genel puan vermektedir. SuperPi stres testi, daha iyi sonuç vermesi için XtremeSystems.org tarafından modifiye edilmiştir. Test çalıştırıldığında, uygulama, pi sayısının istenildiği kadar (örneğin bir milyon) basamağını hesaplamaktadır. Şekil 1.9.' da uygulamanın sonuç ekranı gösterilmiştir.

Prime95; asal sayıları aramak için kullanılan dağıtılmış bir projedir. www.mersenne.org tarafından dağıtılmaktadır. Sonuçları işlem/milisaniye cinsinden vermekte ve doğrudan web sitesine kaydetmektedir. Şekil 1.8.'de uygulamanın test ekranı gösterilmektedir. Uygulamanın internet adresinde kıyaslama sonuçları

görülebilmektedir (Mah Ung ve Case, 2010), (http://www.mersenne.org /report_benchmarks/).



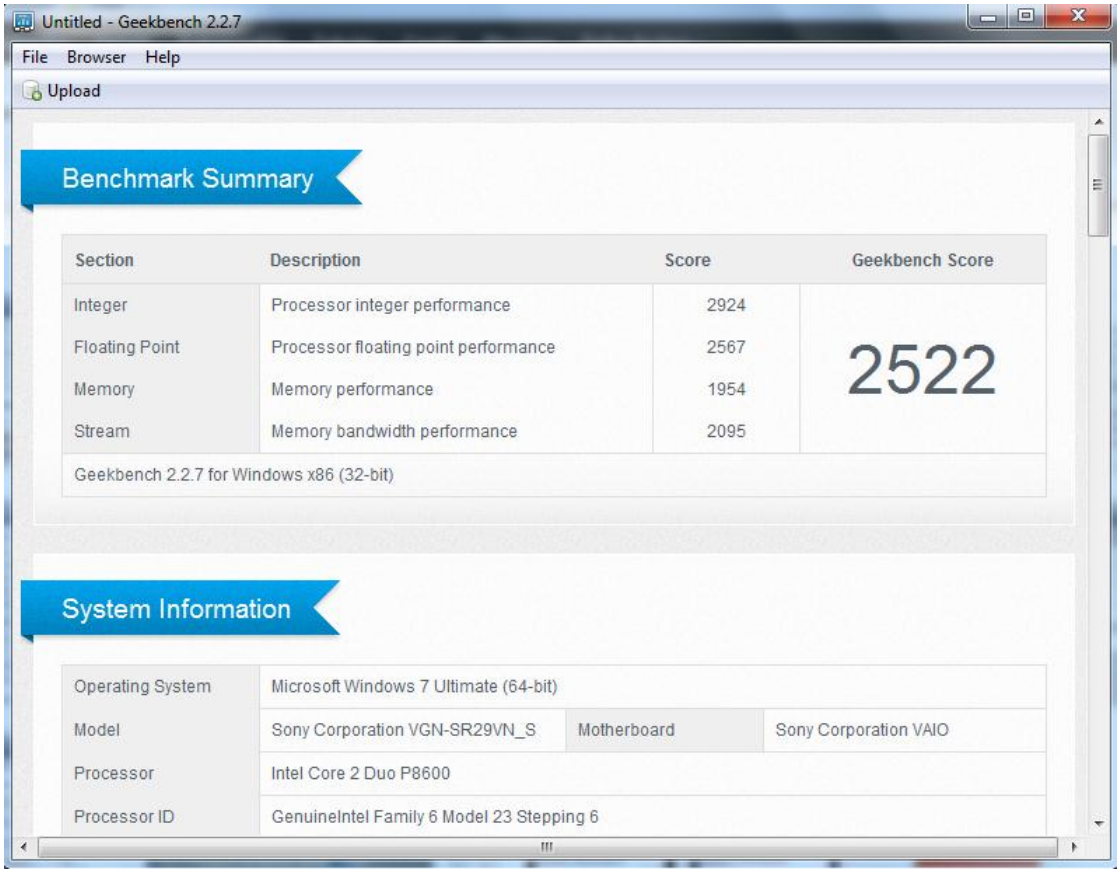
Şekil 1.8. Prime95 programının ekran görüntüsü.



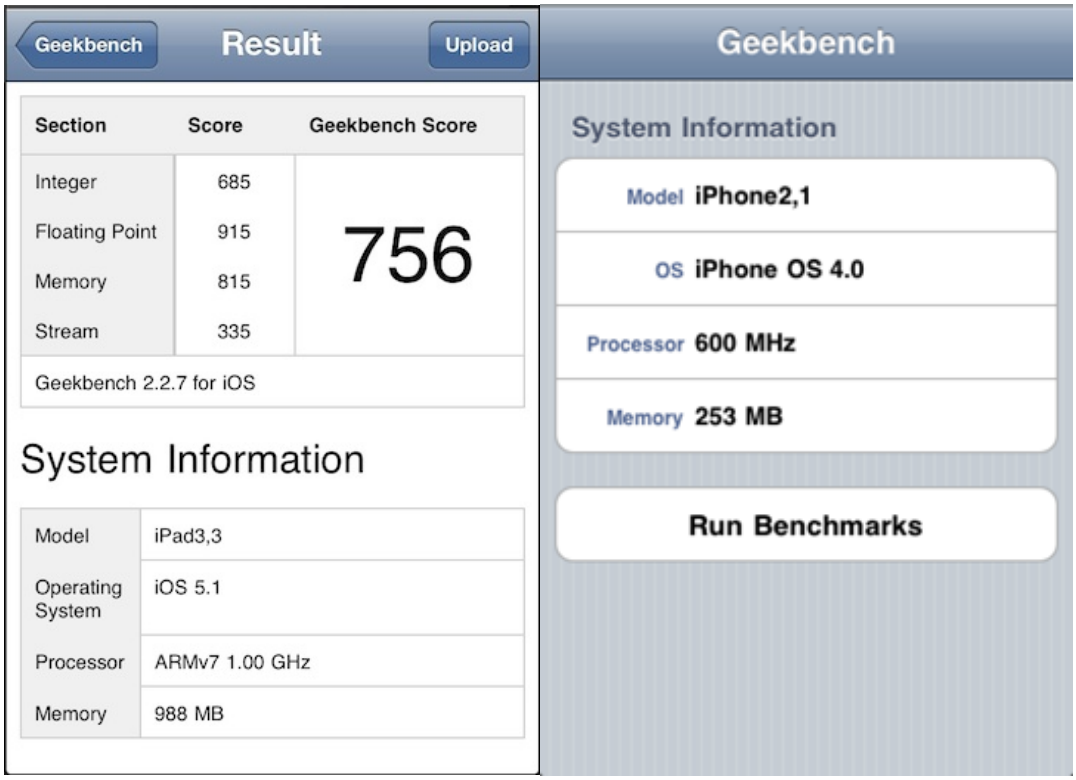
Şekil 1.9. SuperPi programının ekran görüntüsü.

1.7.8. GeekBench

Son dönemin popüler benchmark yazılımı olan GeekBench, diğer yazılımlardan farklı olarak, sadece Windows tabanlı bilgisayarlarda değil, Linux ve Mac bilgisayarlar üzerinde çalışabilmektedir. IOS ve Android işletim sistemli cep telefonu ve tablet bilgisayarlarda da test yapabilmektedir. Bilgisayarın integer, floating point performansını ölçebilmekte, bellekte ayırma işlemi yaparak bellek performansını ölçebilmektedir. Şekil 1.10.' da GeekBench yazılımının PC üzerindeki ekran görüntüsü, şekil 1.11.' de de Android veya ios tabanlı cep telefonu ve tablet pc' lerdeki sonuç ekranı örneği gösterilmektedir.



Şekil 1.10. GeekBench programının ekran görüntüsü.



Şekil 1.11. GeekBench' in, cep telefonu veya tablet pc' deki ekran görüntüsü.

2. KURAMSAL TEMELLER

2.1. Performansı Kavramları

Bilgisayarın hızını belirleyen farklı kavramlar vardır. Bunlardan en önemlisi "işlemci hızı" dır. Kullanıcıya, üretici tarafından verilen performans metrik değerleri, bilgisayarın ne kadar hızlı çalıştığını teorik olarak göstermektedir.

- **Milisaniye (msec) :** bir saniyenin binde biri (0.001 veya 1 x 10-3).
- **Mikrosaniye (µsec) :** bir saniyenin milyonda biri (0.000001 veya 1 x 10-6).
- **Nanosaniye (nsec) :** bir saniyenin milyarda biri (0.000000001 or 1 x 10-9).
- **Hertz (Hz) :** bir saniyedeki çevrim (cycle) sayısı. Aynı zamanda uluslararası frekans metrik değeridir.
- **MegaHertz (MHz) :** Saniyedeki milyon çevrim sayısı.
- **Saat hızı:** Genellikle Mhz ölçüsünde ifade edilen, saniyede yürütülen saat döngüsü sayısıdır.

Bilgisayarın performansın etkileyen faktörler;

- İşlemcide register, integer, kayan nokta ve string işlemleri.
- Önbellek, ana bellek ve dış bellek okuma, yazma, erişim süresi faktörleri.
- Bellek boyutu
- Dosya sistemi
- Derleyici
- Giriş çıkış işlemleri
- Paralel bilgisayarlarda iletişim faktörleri

Bir bilgisayarın performansını ölçmek için, çeşitli görevler üzerinde yürütme zamanı ölçümü yapmak gerekir. Bu işlem aşağıdaki ölçüm birimleri ile ifade edilir;

- **Kullanıcı zamanı (User time):** İşletim sisteminin yükü hariç tutularak, işlemcinin belirli bir işi bitirme süresidir.

- **Sistem zamanı (System Time):** İşletim sisteminin verilen işlemi bitirme süresidir.
- **Geçen zaman (Elapsed Time):** Bir işlemin başlangıcından sonlandırılmasına kadar geçen süredir.
- **Tepki süresi (Response Time):** Bir işlemin başlaması için bir başka işlemin bitmesini beklemesi arasında geçen süredir (Okulicka, 2010).

2.2. Benchmark Uygulama Kategorileri

Benchmarklar, karmaşıklık ve boyut kriterleri ele alınarak; mikrobenchmark ve makrobenchmark olarak iki kategoriye ayrılmaktadır.

2.2.1. Mikrobenchmarklar

Mevcut platform üzerinden çalışan yazılım veya donanım performans testleridir. Bazen mikrobenchmarklar, problem çözebilecek küçük kod parçacıklarını da içerirler. Skorlar, bir zaman birimi üzerinden ölçülerek, programcının tasarlayacağı bir puanlama sistemi üzerinden verilmektedir. Bu puanlama grafikle de ifade edilebilir. Mikrobenchmarklar ile bir sistemin düşük seviye maliyet kıyaslaması yapılabilir, fakat tam geçerli bir fiili işlem kapasitesi ölçümü yapılamaz (Li, 2009).

Microbenchmarking' e kısa bir örnek olarak;

```
D := GetTickCount; // Anlık süre değeri
for I := 1 to 1000 do
for J := 1 to 1000 do
    begin
      FreeMem(A[J]);
      GetMem(A[J],Random(100)+1);
    End;
D := GetTickCount-D;
```

Yukarıdaki kod kümesinde bellekte yer boşaltılıp yer ayrılmakta ve işlemin başında ve sonunda süre ölçümü yapılmaktadır. Bu kodu çalıştırarak basitçe bellek performansına bir puanlama üretilebilmektedir.

2.2.2. Makrobenchmarklar

Genellikle gerçek uygulamalardan oluşan orta ve büyük ölçekli uygulamalardır. Makrobenchmarklar genellikle tipik durumlarla ilişkili 8 veri ile gelir. Benchmark çalıştırıldığında, veri programa girilir ve çıkışa kadarki süre ölçülür. Makrobenchmarklar gerçek uygulamalardan çok daha fazla sistem üzerinde stres oluştururlar. Benchmark sonucunun iyi bir gösterge olarak kullanılabilmesi için testin yapıldığı ortamlarda çalışan uygulamaların birbirinin aynısı olması gerekmektedir (Zhang, 2001).

2.3. Endüstri Standardı

Birçok farklı standart enstitüsü benchmark testlerinde kriterler tanımlamaktadır. Bunlardan en ünlüleri:

- Standard Performance Evaluation Corporation (SPEC)
- Transaction Processing Performance Council (TPC)
- BAPCo (Windows kişisel bilgisayarları için benchmarklar üreten bir endüstriyel birliktir.)
- Synchromesh Computing benchmark
- Gömülü Mikroişlemciler Benchmark Birliği (Embedded Microprocessor Benchmark Consortium (EEMBC))
- Java Micro Benchmark

Diğerleri;

- Khornerstone
- Aquamark

- GL Excess
- BRL-CAD Benchmark

(Moore, 2008)

2.3.1. Standard Performance Evaluation Corporation (SPEC) Kuruluşu

SPEC, çok çeşitli standart kriterler tanımlar. Bu kriterler, ağ dosya sunucularından, yüksek performans bilgisayarlara kadar geçerlidir. Bu kriterler arasında, SPEC CPU benchmark paketi, muhtemelen bilgisayar literatüründen yaygın kullanılan ölçüttür. SPEC CPU, tamsayı paketi (integer suite) ve kayan nokta paketi (floating point suite) olarak ikiye ayrılmıştır (Zhang, 2001).

Standart Performans Ölçüm Kurumu, amacı bilgisayarların başarımlarını ölçebilen dürüst, tarafsız ve anlamlı denektaşları üretmek olan, kâr amacı gütmeyen bir şirkettir. Kuruluş yılı 1988'dir. Üyeleri arasında, önde gelen bilgisayar ve yazılım şirketleri yer almaktadır. Günümüzde SPEC'in değerlendirmeleri, bilgisayar sistemlerinin geliştirilmesinde yaygın olarak kullanılmaktadır. Bu değerlendirmeler kurumun internet sitesinde yayımlanmaktadır. Değerlendirme sonuçları gayri resmi olarak "SPECmarks" veya "SPEC" olarak anılmaktadır. Kurumun değerlendirmelerinde günlük hayatta karşılaşılacak durumlar sınanmaktadır. Bu değerlendirmeler için çeşitli sınama programları kullanılmaktadır. Örneğin SPECweb2005 değerlendirilmesinde internet sunucularının başarımı ölçülmektedir. Bunun için farklı türlerde paralel HTTP istekleri gönderilerek başarım ölçülmüştür. SPEC CPU ise Merkezi İşlem Birimi'nin başarımını farklı türlerdeki programların çalışma zamanlarını ölçerek sınamaktadır. Örneğin bir derleyici ve bir satranç oyunu çalıştırılarak elde edilen sonuçlar değerlendirilmiştir. Bütün bu çalıştırılan görevlere kullanım sıklıklarına göre farklı ağırlıklar verilir. Daha sonra bu ağırlıklar kullanılarak tek bir benchmark değerlendirme sonucu elde edilir.

SPEC benchmarklar, uyumlu sanal makineye sahip, her türlü bilgisayarda çalışabilecek programlama dillerinde (çoğunlukla C yada FORTRAN) yazılmaktadırlar. Benchmarkı kullanacak olan üyeler bu kodu değiştirmeden herhangi bir derleyici ile derlerler ve kullanırlar. Bu kodlar sayesinde üreticiler farklı

SPEC denektaşlarının başarımlarını artırmak için derleyicilerini geliştirmektedirler (Bird vd., 2012).

SPEC, gerçek benchmark uygulamaları geliştirerek önemli sistem bileşenleri bir standart üzerine skorlamak için kurulmuştur. SPEC, 'hangi bileşeni hangi yükle test ederim?' konusu ile ilgilenmektedir. SPEC için asıl soru: "Sistem performansı nedir ve nasıl kıyaslanır" dır. Sistem performansı birçok farklı kriter ile kıyaslanabilir:

- Teknoloji
- Mimari
- I/O alt sistemi
- İşletim Sistemi
- Saat frekansı
- Veri yolu protokolleri
- Derleyiciler
- Kütüphaneler
- Uygulama yazılımı

Ancak, son dönemde çoklu işlemci kullanımı, sistem performans ölçümlerini zorlaştırmıştır (http://en.wikipedia.org/wiki/SPEC).

2.3.2. Transaction Processing Performance Council (TPC) Kuruluşu

1988 yılında kurulmuş ve kâr amacı gütmeyen bir organizasyon olan TPC, doğrulanabilir ve objektif sonuçlar verebilen, veri ve veri tabanı benchmarkları üzerine çalışmaktadır (Nambiar, 2010).

2.3.3. Bap Corporation (Bapco) Kuruluşu

Bapco, kâr amacı gütmeyen, işletim sistemi ve yazılım uygulamalarına dayalı, kişisel bilgisayarları test etmek amacıyla yöntemler geliştiren bir organizasyondur. Windows kişisel bilgisayarları için benchmarklar üreten bir endüstriyel birliktir.

Bapco' nun mevcut üyelerinden en ünlüleri; RCintuition, Atheros Communications, CNET, Compal Electronics, Dell, Hewlett-Packard, Intel, Lenovo, Microsoft, SAMSUNG, SanDisk, Seagate, Sony, Toshiba, VNU Business Publications Limited (UK), ZDNet, ve Ziff Davis' tir (Nambiar, 2010).

2.4. Benchmark Algoritması Türleri

Üç tane popüler benchmark türü vardır, bunlar:

- Çekirdek (Kernel)
- Sentetik (Synthetic)
- Uygulama

Çekirdek benchmarklar genelde kodun ortalama yüzde10'luk bölümünün işlemci kaynaklarının yüzde 80'ini kullandığı analizler üzerine kuruludur. Performans analistleri bu kod parçacıklarını ayıklayıp, benchmark olarak kullanmışlardır. Çekirdek benchmarklara örnek olarak "Livemore Loops" ve "Linpack Benchmarks" gösterebiliriz. Çekirdek benchmarklarının genel zaafları; boyut olarak küçük olmaları ve sadece CPU performansını ölçebilmeleridir.

Sentetik benchmarklar, performans analistinin deneyim ve bilgisine dayalıdır. Sentetik benchmarklara örnek olarak Dhrystone ve Whetstone verebiliriz. Sentetik benchmarklarda da çekirdek benchmarklara benzer sorunlarla karşılaşılabilir. (Weiderman, 1989)

Kullanıcı gözünden bakıldığı zaman, en iyi benchmark yazılımı kullanıcının kendi yazılımıdır. Tescil edilmiş binlerce benchmark yazılımı bulunmaktadır. Çok sayıda testi ardı ardına yapan programlar uzun bir işlem süresi gerektirdiği için çok fazla tercih edilmemektedir.

2.4.1. Dhrystone Benchmark Algoritması

Dhrystone;1984 yılında Reinhold Weicker tarafından geliştirilmiştir. Bu sentetik benchmark string' ler üzerinde ciddi bir zaman harcamaktadır ancak küçük makinelerin integer performansını ölçmek için tasarlanmıştır. Performans ölçüm değeri sonucunu "saniyedeki dhrystone" (Dhrystones per second) olarak vermektedir (Weicker, 1991).

2.4.2. Linpack Benchmark Algoritması

Linpack; bir lineer cebir işlemleri grubu olarak 1976'da Jack Dongarra tarafından tasarlanmıştır. Bilgisayarın kayan nokta performansını ölçmekte kullanılmaktadır. Bu benchmark, 100x100 bir matris kullanmaktadır. Çoğu işlemcinin ön belleğine sığabilecek kadar küçük bellek işgal etmektedir. Benchmark sonucunu saniyede …milyon kayan nokta işlemi (by millions of floating-point operations per second (MFLOPS)) olarak vermektedir. Performans ölçümleri tek ve çift hassasiyetli olabilir (Dongarra, 1983).

2.4.3. Whetstone Benchmark Algoritması

Whetstone;1976 yılında H.J. Curnow ve B.A. Wichman tarafından yazılmış popüler bir sentetik kayan nokta ölçütüdür. Bu benchmark, çeşitli sayısal hesaplamalar (diziler, trigonometrik fonksiyonlar gibi) gerçekleştiren modülleri içerir. Benchmark, küçük ve hassas modüllerden oluşur ve ön bellek boyutuna duyarlıdır. Performans ölçüm sonuçları "saniyede tek ve çift hassasiyetli Whetsone" (single- and double precision Whetstones per second) olarak verilir (Curnow vd., 1976).

2.4.4. Whetstone ve Dhrystone Algoritmalarının Avantajları

- İyi tanımlanmış ve birkaç farklı dilde uygulanmaktadırlar.

- Her 1000 Whetstone ve Dhrystone modern bilgisayarlarda sadece birkaç milisaniye sürer. Bu, 100hz frekans civarında ölçüm yapabilmeyi mümkün kılar (York, 2002).
- Whetstone, bilimsel programlara karakteristiğine, Dhrystone sistem programlama karakteristiğine sahiptir.

2.4.5. Whetstone ve Dhrystone Algoritmalarının Dezavantajları

- Çok küçük boyutludurlar.
- Kullanılmayan kod fazlalığı vardır.
- Kontrolsüz kaynak kodu barındırırlar.
- Derleyici hilelerine dayanıksızdırlar.
- Modern makinelerde çok kısa çalışma zamanına sahiplerdir.
- Tek bir kriter ile performans karakterizasyonu (Hartstone, 1989).

Bir bilgisayar sisteminin performansı tek bir sonuç veya tek bir kriter ile karakterize edilemez. Tüketici birçok performans ölçüm programının sonuçlarını inceleyip, hiç anlamadığı terimlerle karşı karşıya bırakılmaktadır. Ne yazık ki üretici firmalar da, kendi ürünlerini ön planda tutabilmek için tescilli veya tescilsiz çeşitli benchmark kullanarak, bunlardan iyi sonuç aldıklarını tüketiciye sunmaktadırlar. Bu tüketicinin ürünler arasında karşılaştırma yapmasını ve seçme şansını zora sokmaktadır.

2.5. Performans Hesaplanması Yöntemleri

Performans ve toplam çalışma zamanı, bilgisayarda ters orantılıdır. Çalışma zamanı azaldıkça performans artar. Aşağıdaki ifade ile;
Performans X = 1 / Çalışma zamanı X
Eğer Bilgisayar X, Bilgisayar Y'den n kat daha hızlı ise,
PerformansX / Performans Y = Çalışma zamanıY / Çalışma zamanı X = n
Çalışma zamanı : Başlangıçtan bitişe kadar toplamsüre (disk erişimi, bellek erişimi, giriş çıkış aygıtları için geçen süre, işletim sistemi vs.)

CPU Zamanı: CPU'nun işlemi bitirene kadar geçirdiği zaman. (Bu zamanın içinde, giriş çıkış aygıtları için bekleme, işletim sistemi işlemleri ve diğer programlara bağlı süreler hesaba katılmaz.)
Kullanıcı CPU zamanı: Program için harcanan CPU zamanı
Sistem CPU zamanı: Program için işletim sisteminin içinde harcanan toplam zaman

Bilgisayarlar belirli bir sabit oranda çalışan bir donanımsal saat kullanılarak tasarlanmıştır. Bu saat kesikli ve kısa zaman aralığı ile çalışır. Saat hızı ile saat periyodu ters orantılıdır.

Programın CPU çalışma zamanı: Programın CPU saat döngüsü * zaman saat döngüsü (Sheth, 2005)

3. MATERYAL VE YÖNTEM

3.1. Materyal

Uygulama geliştirilirken, programlama dili olarak Borland Delphi yazılımının 7. versiyonu, veri tabanı uygulaması olarak, Borland Database Engine kullanılmıştır. Program ara yüzünün arka planının oluşturulması ve diğer resimlerin düzenlenmesinde Paint Shop Pro 5 kullanılmıştır. Programın kurulum paketinin hazırlanmasında Install Shield Express kullanılmıştır.

Programın çalıştırılması ve test aşamasında 6 farklı bilgisayar kullanılmıştır. Programın güvenilirliğinin test edilmesinde, Primate Labs firmasının geliştirdiği GeekBench yazılımının 2.2.7 versiyonu kullanılmıştır.

3.1.1. Borland Delphi

İlk sürümü Şubat 1995'te satışa sunulmuş olan, temel programlama dili Pascal olan uygulama geliştirme programıdır. Nesne, sınıf, fonksiyon, aşırı yükleme gibi NYP tekniklerini ve daha fazlasını içermektedir (http://tr.wikipedia.org/wiki/Delphi_%28 programlama_dili%29).

Uygulama geliştirmede kullanılmış olan Delphi bileşenleri;

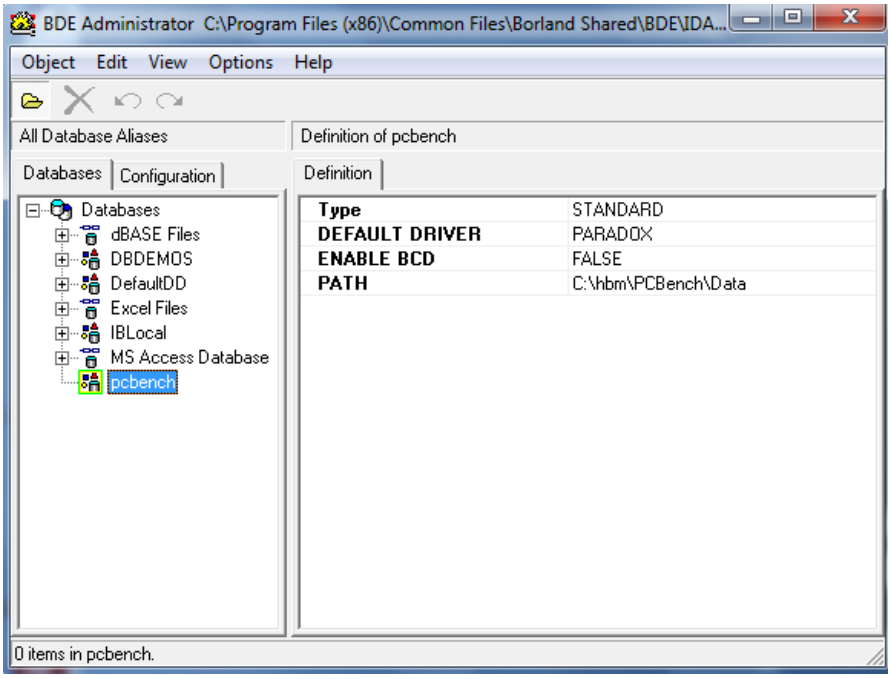
- **Button:** Üzerine tıklayınca "onclick" fonksiyonunu çağıran Delphi komponentidir.
- **Combobox:** "Items" özelliğinin içeriğindeki string' leri alt alta listeleyen komponenttir. Uygulamada genelde, üzerine tıklandığında "onclick" fonksiyonunu çağırması kullanılmıştır.
- **Gauge:** Sıfır değeri ile, girilen "Maxvalue" (en yüksek) değer arasında, "Progress" özelliğinin sayısal büyüklüğü kadar çizgisel grafik çizen komponenttir. Sonuç grafiklerinde ve CPU, Bellek, HDD doluluk oranlarının hareketli grafik olarak gösteriminde kullanılmıştır.

- **Image:** “Picture” özelliğine gönderilen string ile belirtilen .bmp dosyasını forma yükleyen komponenttir.
- **Label:** Çalışma zamanında “Caption” özelliğine gönderilen string değerini form üzerinde gösteren komponenttir.
- **QuickReport:** Delphi’ de, üzerindeki bileşenlere gönderilen değerleri, yazıcıdan çıktı alınabilecek şekilde form oluşturan komponenttir. Uygulamada, benchmark sonuçlarının yazıcıdan çıktı olarak alınabilmesi için kullanılan raporlama komponentidir. QRLabel, QRImage, QRBand vb. gibi alt bileşenlere sahiptir. Uygulamada tam test sonucunda verilen yazdırılabilir sayfanın oluşturulmasında kullanılmıştır.
- **Table:** Veri tabanına, veri tabanı dosyalarına ve dosyaların içerisindeki kayıtlara erişmek, eklemek, silmek ve düzenlemek için kullanılan komponenttir.
- **Timer:** “Interval” özelliğinin sayısal büyüklüğü kadar milisaniye değerine göre işlem yapan komponenttir. Her interval milisaniyesinde bir kez “OnTimer” fonksiyonunu çağırmaktadır.

3.1.2. Borland Database Engine

Çalışmakta olan programda, kullanıcı verilerinin belirli bir düzen içinde kaydedilmesi ve erişimi işlemlerini gerçekleştiren yazılımlara veri tabanı denir. Veri tabanı dosyalarına, dosyalardaki kayıtlara erişimi sağlayan ve bu kayıtlar arasında ilişkilendirme sağlayan yazılımlara veri tabanı motoru (database engine) denir.

Borland Database Engine; Borland yazılım geliştirme araçları için tasarlanmış bir veri tabanı motorudur. Veri tabanı oluşturmak, farklı bilgisayarlarda veri tabanına erişim sağlamak için gerekli dosyaları sağlamak için kullanılan araçtır. Şekil 3.1.’ de BDE kullanılarak oluşturulan alias ekran görüntüsü verilmiştir.



Şekil 3.1. BDE ile alias oluşturma.

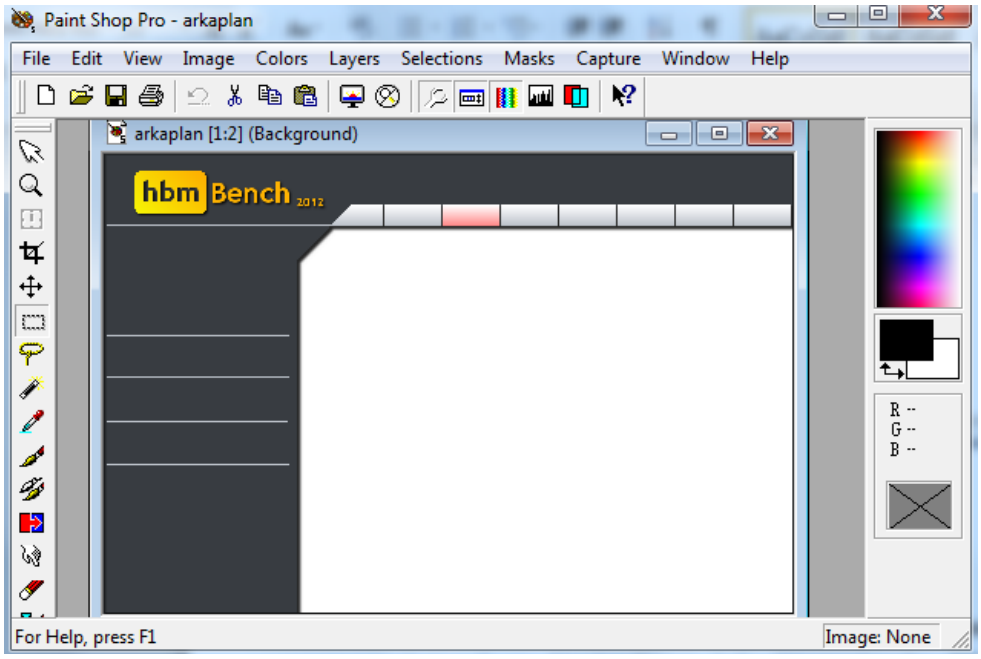
3.1.3. Borland Database Desktop

Borland Database Desktop; BDE ile oluşturulan veri tabanında yeni dosya oluşturmak ve düzenlemek için kullanılan uygulamadır. BDE ile veri tabanı dosyalarında, izin verilen kayıt formatlarında alanlar oluşturulup ilişkilendirme yapılabilmektedir.

3.1.4. Jasc Paint Shop Pro

Jasc Corporation' ın geliştirdiği resim düzenleme yazılımıdır. Adobe firmasının Photoshop yazılımının rakibidir. PSP 9 sürümünden sonra Corel firmasına satılmış olup yeni sürümler Corel firması tarafından geliştirilmektedir. PSP, pekçok webmaster tarafından web sitesi tasarımında resim düzenleme yazılımı olarak kullanılmaktadır (http://tr.wikipedia.org/wiki/Paint_Shop_Pro).

Uygulamada PSP' nin 5. versiyonu; arayüz arka planının hazırlanması, marka logolarının boyutlandırılması ve kullanılan diğer resimlerin oluşturulması ve düzenlenmesinde kullanılmıştır (Şekil 3.2.).



Şekil 3.2. Programın arka planının tasarımı.

3.1.5. Install Shield Express

Program dosyalarını, veri tabanı motorunu ve kısa yolları hedef bilgisayarda kuracak şekilde, uygulamaya bir kurulum paketi (CD, disket vb.) hazırlamak için kullanılan uygulamadır. Install Shield Express ile hazırlanan paket, "Setup" adlı dosyayı çalıştırılarak, geliştirilen yazılım hedef bilgisayara kurulabilmektedir.

3.1.6. Uygulama Test Platformu

Uygulama geliştirildikten sonra, sayıca 30' un üzerinde masaüstü ve dizüstü bilgisayarda denenmiştir. Bu bilgisayarlardan rasgele 6 tanesi istatistiki bilgi vermek ve uygulamanın farklı benchmark uygulamalarıyla kıyaslanması için kullanılmıştır;

- SONY VAIO SR29VN Notebook (Intel P8600 CPU, 4 GB 800Mhz RAM, 256MB ATI HD3470 GPU)
- HP Compaq CQ61 Notebook (Intel Dual Core T4300 CPU, 2 GB 667Mhz RAM, 512MB Nvidia G103M GPU)
- Masaüstü PC (Intel Core i5 2400, 8 GB 1333 Mhz RAM, 1 GB GeForce GTX460 GPU)
- Exper Notebook (Intel Core i3 M370, 2GB DDR3 1333Mhz, Intel HD Graphics GPU)
- Packard Bell NEW91 Notebook (Intel Core i3 M350 CPU, 3GB DDR3 1333Mhz RAM, 1GB Intel GT320M GPU)
- Masaüstü PC (AMD X2 250 CPU, 2 GB 1333Mhz RAM, Nvidia Entegre GPU)

Uygulamanın başarımının kıyaslanması için baz alınan yazılım olan Geekbench; Primate Labs firması tarafından yazılmış, "tek tıkla benchmark" prensibi ile çalışan, hızlı algoritmalar ile hızlı sonuç verebilen, tek ve çoklu çekirdek ayrımı yapabilen, 64 ve 32 bit işletim sistemlerinde çalışabilen etkili ve popüler bir benchmark yazılımıdır. Geekbench' i diğer uygulamalardan en önemli özelliği, sadece Microsoft Windows değil, hem MacOS, iOS, Linux, Android işletim sistemlerinde de çalışabiliyor olması hem de sadece PC değil, MAC, cep telefonu ve tablet PC' ler üzerinde de çalışabiliyor olmasıdır.

3.2. Yöntem

PC Bench uygulaması, Borland Delphi 7 yazılım geliştirme platformunda, çeşitli nesneler ve işlemler kullanılarak yazılmıştır. İşlemci, RAM, HDD ve grafik kartı bileşenlerinin test edileceği algoritmalar seçilmiştir. Algoritmalar, öncelik sıralamasına alınmış ve her algoritma, test ettiği bileşene göre sınıflandırılmıştır. Algoritmalar, Borland Delphi 7' de, her biri farklı birer çalıştırılabilir program dosyası (.exe) olacak şekilde geliştirilmiş, her algoritma çalışma zamanı ve bellek kullanım miktarı gibi verileri, Log klasörü altına, metin dosyaları biçiminde kaydedecek şekilde programlanmıştır. Paint Shop Pro 5 yazılımı ile, ana program

modülünün ara yüzünün arka planı tasarlanmıştır. Arayüz, paneller ile modüllere ayrılmış, her modülün ayrı ayrı çalışabilmesi sağlanacak ayırıcı fonksiyonlar kullanılmıştır. Geliştirilen benchmark algoritmaları arayüze bağlanmış, benchmark sonuçlarının grafiksel ifadeleri ve puanlama hesapları gerçekleştirilmiştir. İşlemci, HDD, RAM, anakart, grafik kartı ve diğer PC bileşenlerinin WMI fonksiyonları ile uygulama arayüzünde bilgi amaçlı gösterimi yapılmıştır. Bilgisayar bileşenlerinin markalarına göre logolarının arayüzde gösterimi gerçekleştirilmiştir. Test sonuçlarının hem QuickReport ekranında yazıcı dökümü alınabilecek formatta gösterimi yapılmış, hem de bu sonuçlar, kullanıcının kıyaslama yapabilmesi için grafiksel ifadelerin olduğu bir modülde gösterilmiştir. Test sonuçlarını kaydedip farklı sonuçlarla kıyaslama yapabilecek biçimde veri tabanı hazırlanmıştır. Veri tabanı motoru olarak BDE kullanılmış, veri tabanı dosyası Borland Database Desktop ile hazırlanmıştır. Install Shield Express ile uygulama ve veri tabanı kurulum paketi hazırlanmıştır. Uygulama altı farklı bilgisayarda Primate Labs GeekBench yazılımı ile aynı anda denenmiş, iki programın benchmark skorları bu bilgisayarlar üzerinde kıyaslanarak, geliştirilen uygulamanın güvenilirliği test edilmiştir.

3.2.1. Program Geliştirme

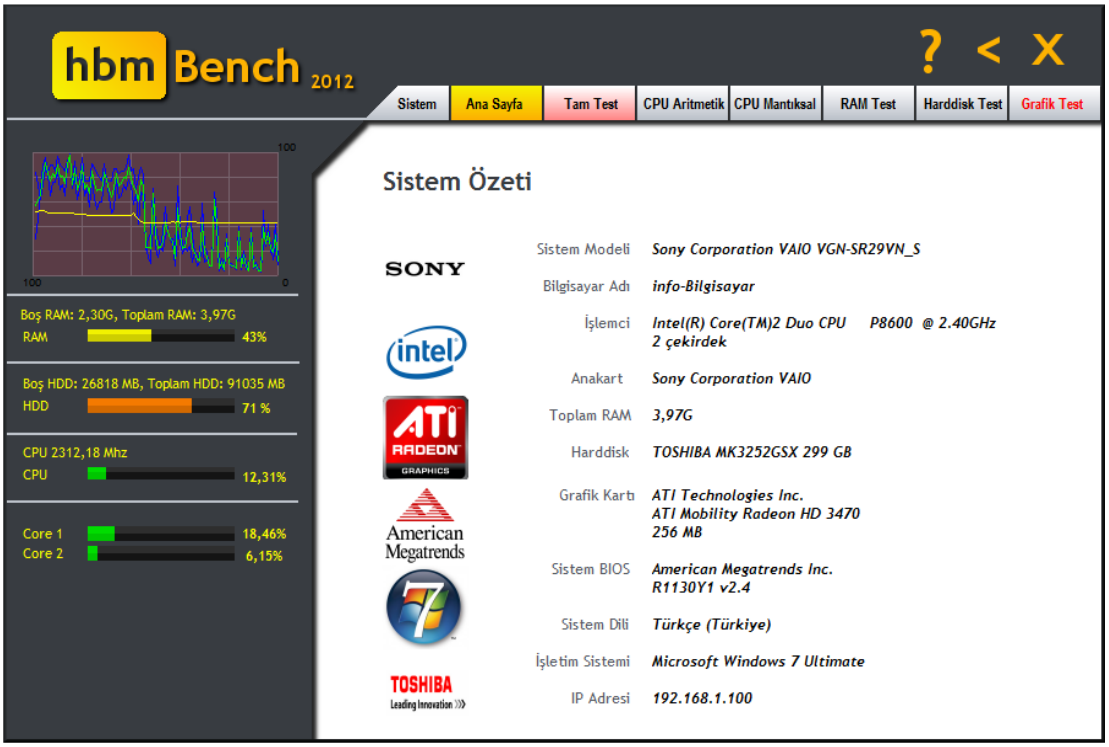
Benchmark programı, çok sayıda algoritmanın, test edilen donanım bileşenine göre sırasıyla çalıştırılmasını sağlar. Her bileşen, çalıştırılan algoritmaların, başlangıcından bitişine kadar geçen süreyi ölçer, bu süreyi bir standart sapma değeri ile çarparak, sonuca göre bileşene puan vermektedir.

Benchmark uygulaması, puanlama işlemini donanım bileşenlerine göre ayrı ayrı yapmak için farklı modüllerden oluşur.

Program geliştirme aşamasında edinilen amaç, kullanıcıyı karmaşık terimlerden uzak tutmak, tek tuşla tüm benchmark algoritmalarını çalıştırıp detaylı skorları kullanıcıya sunabilecek bir arayüz oluşturmaktır.

3.2.1.1. Programın Arayüzünün Geliştirilmesi

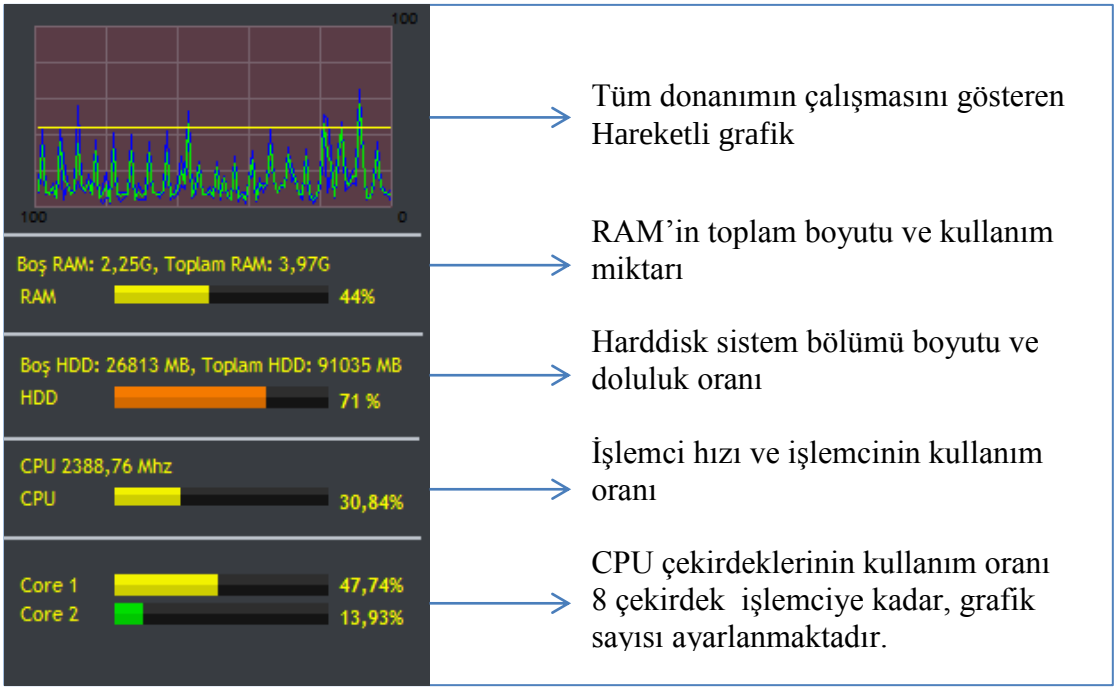
Uygulamaya görsellik katmak için, Paint Shop Pro programı ile arka plan resmi oluşturulmuş, diğer komponentler bu arka plan üzerine yerleştirilmiştir. Uygulamanın ana ekranında 8 adet sekme yerleştirilmiştir. Şekil 3.3.' te ana ekranın sol tarafında görülen grafikler, Şekil 3.4.' te ifade edildiği üzere, sistem bileşenlerinin çalışma yüzdelerini, her saniye yenilenecek biçimde göstermektedir.



Şekil 3.3. Uygulamanın çalışır halde iken ana ekranının görüntüsü.

Sistem: Bu ekranda bilgisayarın donanımları, detaylı özellikleriyle beraber listelenmektedir. Bazı bilgisayarlarda WMI fonksiyonları kullanılamadığı için bu ekran pasif kalabilmektedir.

Ana Sayfa: Bu ekranda; sistemin anakartı, bilgisayar adı, bellek, CPU, grafik kartı, BIOS, işletim sistemi gibi önemli özelliklerin bilgileri verilmektedir. Markalara göre sol tarafta logolar dosyadan aktarılarak gösterilmektedir.



Şekil 3.4. Uygulamanın Timer ile saniyelik performans ölçümlerini grafikle gösterdiği bölüm.

Tam Test: Bu sekmeye tıklandığında, uygulama, bütün test algoritmalarını sırasıyla çalıştırmaktadır. Bu bölümde amaç bilgisayarın tüm testlerini (CPU Aritmetik, CPU Mantıksal, RAM, HDD, Grafik) arka arkaya yaparak, tam bir skor tablosu oluşturmaktır. Bu testin sonunda, mevcut bilgisayar daha önceden veri tabanında kayıtlı değilse veri tabanına kaydedilmektedir. İşlem sonunda kullanıcıya sunulan rapor penceresinde bilgisayarın ayrıntılı skorları gösterilip, yazıcıdan çıktı verilebilmektedir.

CPU Aritmetik Testi: Kullanıcı bu sekmeye tıkladığında, 8 adet algoritma sırasıyla çalıştırılır. Her algoritmanın sonucunda kullanılan bellek miktarı ve test süresi tabloda gösterilmektedir.

- Asal Sayı Testi
- Matris Çarpım Testi
- Text Zip Testi
- Text Unzip Testi

- Sıralama Testi
- Imaj Zip Testi
- Imaj Unzip Testi
- Bmp'den Jpg' ye dönüştürme Testi

CPU Mantıksal Testi: Kullanıcı bu sekmeye tıkladığında, sırasıyla CPU mantıksal test algoritmaları çalıştırılmaktadır;

- Stream Cipher Kriptoloji Testi
- İmaj sharpen Testi
- Gaussian Blur Testi
- Mandelbrot Testi
- BMP Enkript Testi

RAM Testi: RAM' de yazmaç ayırıp o yazmaçlara rasgele veya ardışık olarak veri yazarak çalışan algoritmanın, farklı boyutlarda 16 kez çalıştırıldığı ekrandır. Algoritmanın sonunda, kullanılan bellek miktarları ve süreler tabloda listelenmekte, süreye göre skor üretilmektedir.

Harddisk Testi: Bu ekran altı adet testten oluşmaktadır. Önce harddisk' e 16KB'dan başlayarak, her veri bir öncekinin iki katı olacak şekilde dosya boyutu 32MB'a ulaşana kadar yazma işlemi, ardından aynı şekilde okuma, kopyalama ve silme işlemi yapılır. Sonraki aşamada harddisk' e bloklar halinde veri yazılarak 1 GB'lık dosya oluşturulmakta ve aynı şekilde bloklar halinde tekrar silinmektedir.

Grafik Testi: Kullanıcı bu sekmeye tıkladığında, amacı grafik kartı performansını ölçebilmek olan yedi algoritma sırasıyla çalıştırılır. Bu yedi algoritma; 2D ve 3D çizim fonksiyonları kullanılarak yazılmıştır.

- Rasgele Canvas
- Pikselize
- Döşeme

- Screencapture
- Negatif imaj
- OPENGL FPS Test
- Yeniden boyutlandırma

3.2.2. CPU Aritmetik Test Algoritmaları ve Uygulanması

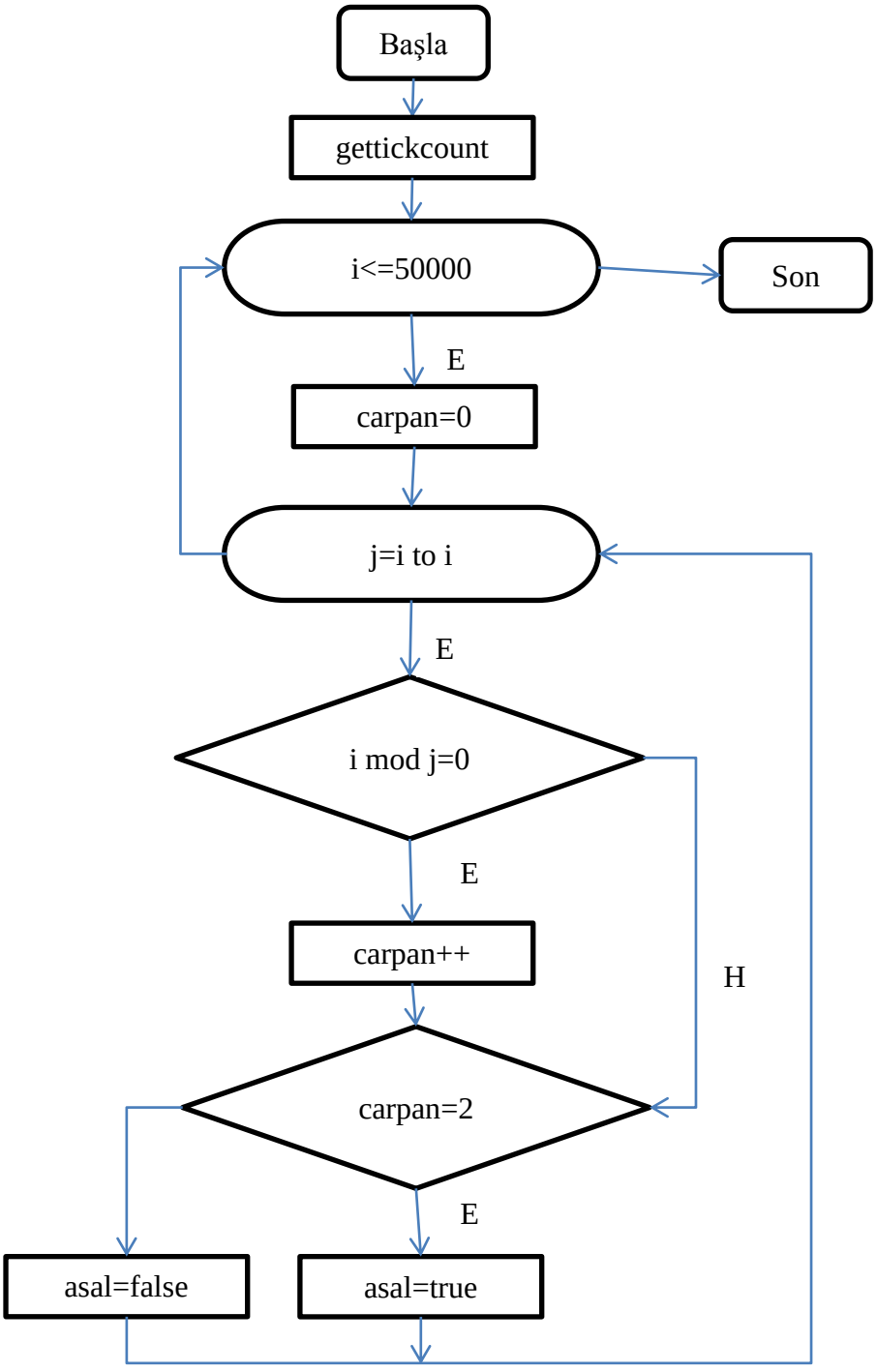
Aritmetik test aşamasında 8 farklı benchmark algoritması uygulanmaktadır. Her algoritma tamamlandıktan sonra ürettiği sayısal skoru bir log dosyasına kaydetmekte ve ana program modülü, bu log dosyalarından skorları okumaktadır. Aritmetik testlerin sonucunda Şekil 3.5' teki gibi sonuçlar gösterilmektedir.

TEST	SKOR	SÜRE	BELLEK	ORTALAMA SKOR
Asal Sayı	1920	521,1 ms	5764 kb	3763
Matris Çarpım	2638	379,1 ms	5924 kb	
Text Zip	4000	25 ms	6832 kb	
Text Unzip	4588	10,9 ms	6776 kb	
Sıralama	4787	40,6 ms	6880 kb	
İmaj Zip	3695	208,9 ms	5776 kb	
İmaj Unzip	5320	9,4 ms	6836 kb	
BMP den JPG ye	3152	1586,5 ms	6600 kb	

Şekil 3.5. Programın, SONY VAIO SR29VN notebook bilgisayarda aritmetik test sonucu gösterimi. (Intel P8600 2 çekirdek CPU, 4GB RAM, ATI 3470 GPU)

3.2.2.1. Asal Sayı Benchmark Testi

Asal sayı bulma algoritması, neredeyse tüm CPU benchmark yazılımlarında kullanılan basit ve etkili bir algoritmadır. İstenilen sayıda üst üste döngü ve kıyaslama işleminden oluştuğu için işlemci üzerinde stres oluşturan bir algoritmadır.



Şekil 3.6. 1’ den 50000’ e kadar asal sayıları bulan algoritmanın akış şeması.

Asal sayı bulan benchmark algoritması, şekil 3.6.’ daki akış şemasında ifadede edildiği şekilde, uygulamada aşağıdaki kod satırları ile oluşturulmuştur;

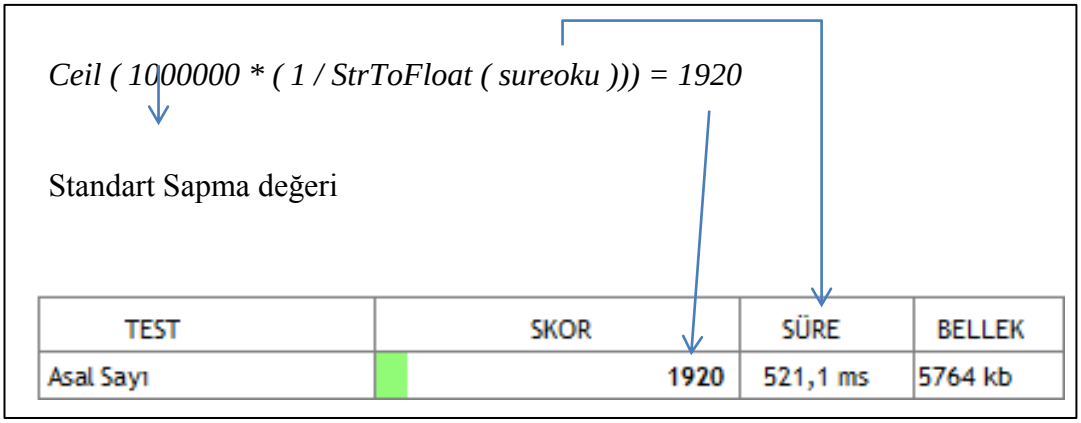
```
gecensure:=gettickcount; // Süreyi kaydet                    (t)
for i:=1 to 50000 do // 1 den 50,000'e kadar asalları bulunsun
begin
        carpan:=0;
        For j:=1 To i Do
                If (i mod j=0) Then carpan:=carpan+1;        (n)
                If carpan=2 Then asal:=True
                else asal:=false;
        end;
gecensure:=(gettickcount - gecensure); //Süreyi kaydet       (t)
```

Büyük O notasyonuna göre bu algoritmanın çalışma zamanı;

O(N)=n+t+t
O(N)=n

"t" (satır tek işlem) değerleri büyük O notasyonunda göz ardı edildiği için algoritmanın hızı "n" olarak gösterilecektir. Algoritma dizi üzerinde işlem yapmadığı için her çalışmada aynı performansı gösterir, hızı logaritmik olarak değişmez.

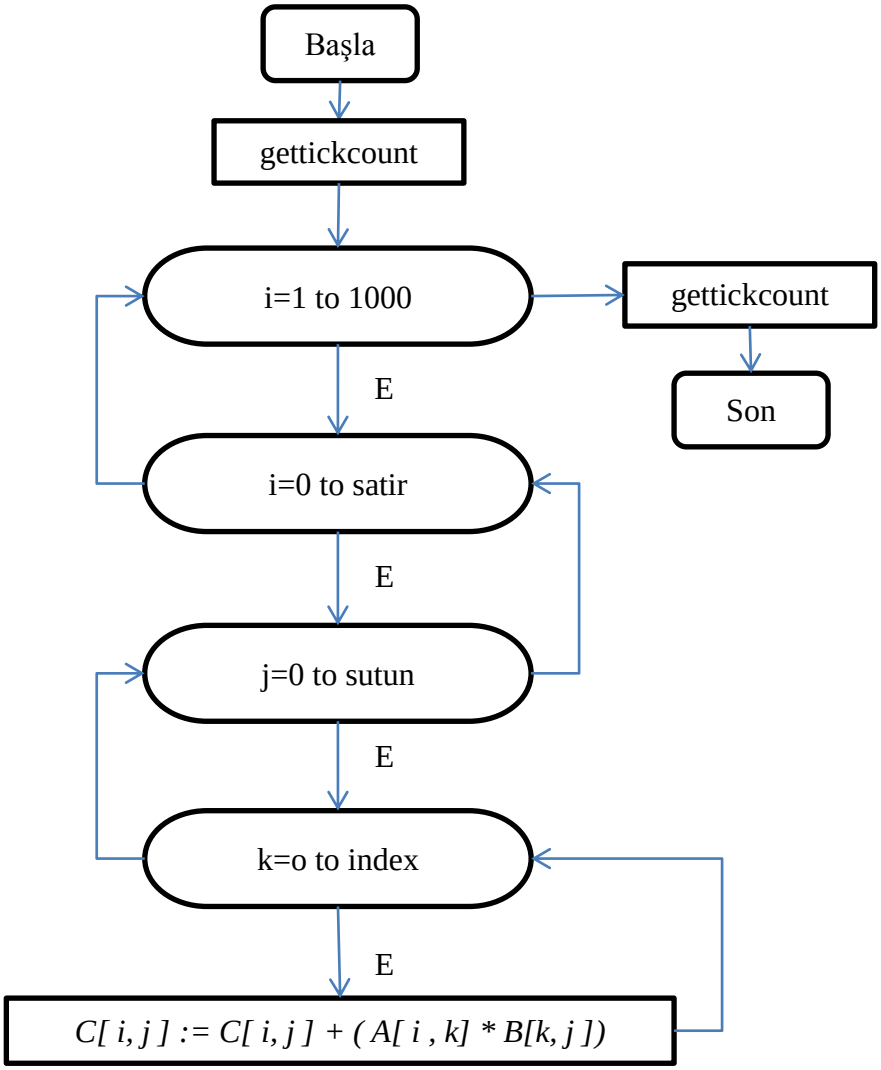
Uygulamada skor hesaplaması yaparken skor'u 0-20000 değerleri arasında tutmak için her algoritmada bir standart sapma değeri belirlenmiştir. (Şekil 3.7.).



Şekil 3.7. Standart sapma değeri kullanılarak 0-20000 arasında skor oluşturulması.

3.2.2.2. Matris Çarpım Testi

Matris çarpım da, asal sayı bulma gibi işlemci üzerinde stres oluşturabilecek, iç içe tekrarlanan döngüler ve matematiksel işlemlerden oluşan bir aritmetik algoritmadır.



Şekil 3.8. Matris çarpım algoritmasının akış şeması.

Şekil 3.8.' deki akış şemasına uygun olarak, uygulamada kullanılmış olan, iki matrisi 1000 defa çarpan kod satırı aşağıda gösterilmiştir;

```
for sayac:=1 to 1000 do                                          (n)
begin
        for i := 0 to satir do                                   (n)
                for j := 0 to sutun do                           (n)
                        for k := 0 to index do                   (n)
                        C[ i, j ] := C[ i, j ] + ( A[ i , k] * B[k, j ]);
End;
```

Büyük O notasyonuna göre algoritmanın çalışma zamanı;

$O(N)=n*n*n*n*t$
$O(N)=n^4$

olarak hesaplanır. Son döngü içindeki satırın çalışma zamanı "t", büyük O notasyonuna göre ihmal edilmektedir.

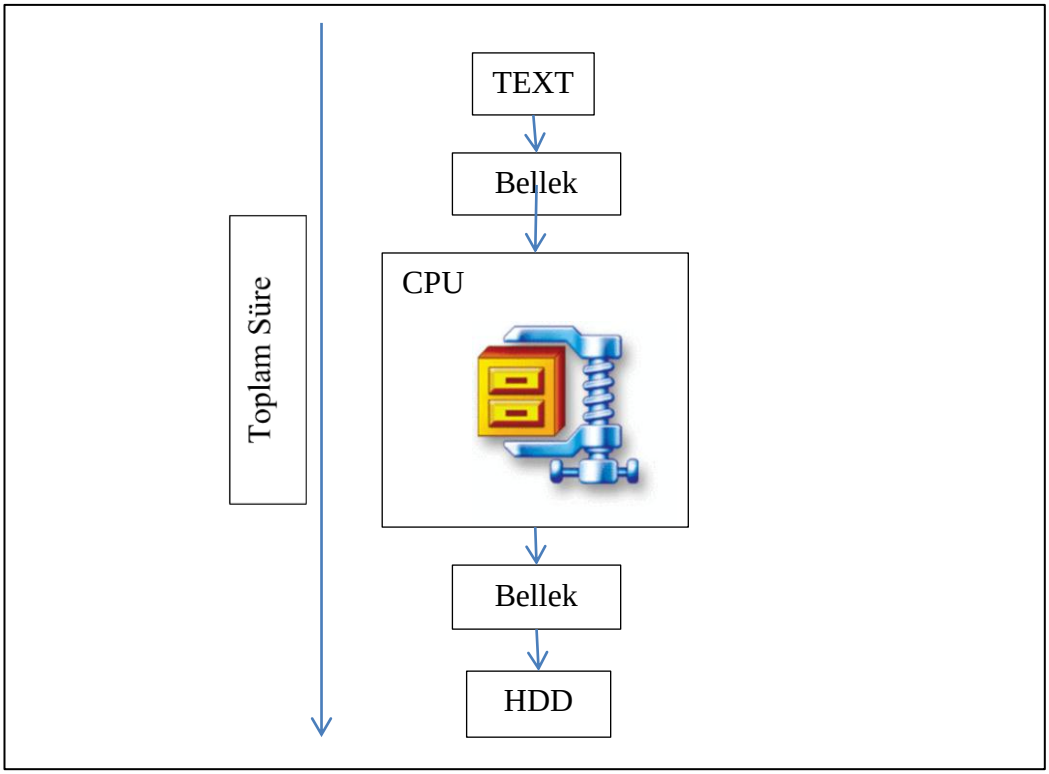
Kod satırında matrislerin 1000 kez çarpılması, günümüzde kullanılmakta olan ortalama hızda işlemciler için 1 saniyeden daha kısa sürmektedir. Çarpım sayısını arttırmak süreyi de aynı oranda arttırmakta ama test sonucunda hesaplanan işlem/saniye değerini değiştirmemektedir. Çünkü bu algoritma işlem hızı, notasyona göre logaritmik değil, n^4 şeklinde hesaplanmaktadır. Amaç, daha az sürede daha çok işlem yapabilmek olduğu için, çarpım sayısı 1000 olarak kullanılmıştır.

3.2.2.3. Text Zip Testi

Zip; bir dosya uzantısı olmakla beraber, bir veya birkaç dosyanın bir paket haline sıkıştırılması ile oluşan dosyalara verilen isimdir. Zip dosya uzantısı, ilk olarak Phil Katz tarafından yazılan PKZIP adlı zip formatlı sıkıştırma ve arşivleme yapan programı ile kullanılmaya başlanmıştır. Microsoft'un Windows Vista ve 7 adlı işletim sistemlerinin kütüphane dosyalarının altında, 'zip' (sıkıştır) ve 'unzip' (çöz) işlemlerini yapan dll kütüphane dosyaları bulunmaktadır. Uygulamada, benchmark için Zip algoritmasının kullanılması, şekil 3.9.' daki sırayladır.

Uygulamada, zip32.dll ve unzip32.dll adlı açık kaynak kod hazır kütüphane dosyaları kullanılmıştır. Aşağıdaki Delphi kod kümesi kısaca zip işlemini yapmaktadır;

```
gecensure:=gettickcount;// Süre hesapla
lboFilesToZip.Items.Add('c:\hbm\PCBench\Temp\text.txt'); // Ziple
ZipFiles('c:\hbm\PCBench\Temp\text.zip', lboFilesToZip.Items);// Zip'i kaydet
gecensure:=(gettickcount - gecensure);// Süre hesapla
```



Şekil 3.9. Zip ve Unzip algoritmalarında sürenin hesaplanması.

3.2.2.4. Text Unzip Testi

Unzip32.dll kütüphanesinin yardımı ile, bir önceki algoritma ile sıkıştırılmış olan "text.zip" dosyasını çözerek text.txt dosyasını elde edilmektedir. Text Zip' teki grafiğin tam tersi işleyiş söz konusudur. Aşağıdaki kod satırı unzip işlemini yapmakta olan kod satırının parçasıdır;

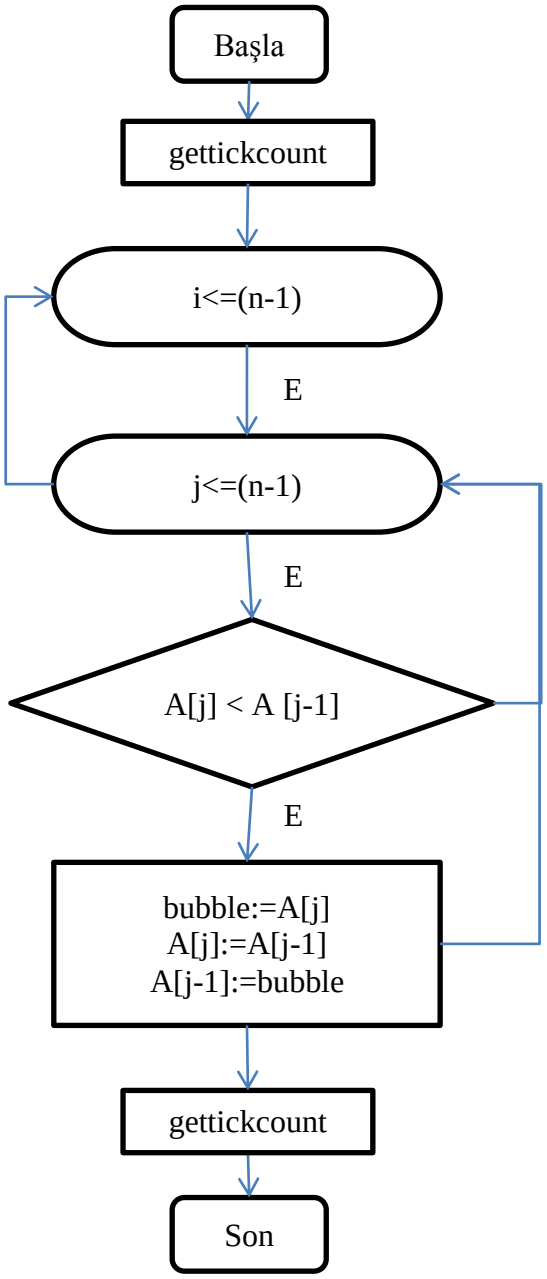
```
edtFileToUnzip.Text := 'c:\hbm\PCBench\Temp\text.zip';
edtUnzipToDir.Text := 'c:\hbm\PCBench\Temp\';
```

3.2.2.5. Sıralama Testi

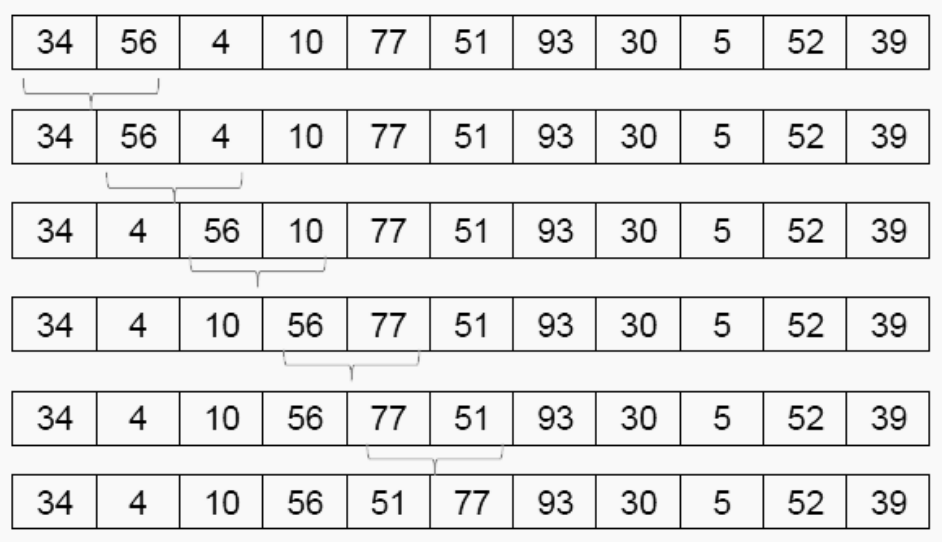
Bu testte; 0-10000 arasında rasgele sayısal değerlerden oluşan on bin haneli karakter dizisi tanımlanmaktadır. On bin haneli bu dizi, 'insertion sort', 'bubble sort' ve

‘quicksort’ adlı çok kullanılan üç farklı algoritmanın her biri ile 10’ar kez sıralanmaktadır.

- Bubble Sort (kabarcık sıralama) algoritması; son derece basit fakat verimsiz bir sıralama algoritmasıdır. Algoritmada, dizinin tüm elemanlarını sırayla ele alır. Ele aldığı her elemanla kendisinden sonraki elemanı karşılaştırır. Bu eleman bir sonraki elemandan büyükse yerlerini değiştirir, küçükse olduğu gibi bırakır ve bir sonraki elemana geçer (Şekil 3.11.). Bu işlem (n-1) kere tekrarlanır. Şekil 3.10.’ da kabarcık sıralama algoritmasının akış şeması gösterilmiştir.



Şekil 3.10. Kabarcık sıralama algoritması akış şeması.



Şekil 3.11. Kabarcık sıralama algoritmasının örnek işleyişi.

Büyük O notasyonu ile kabarcık sıralama incelenirse;

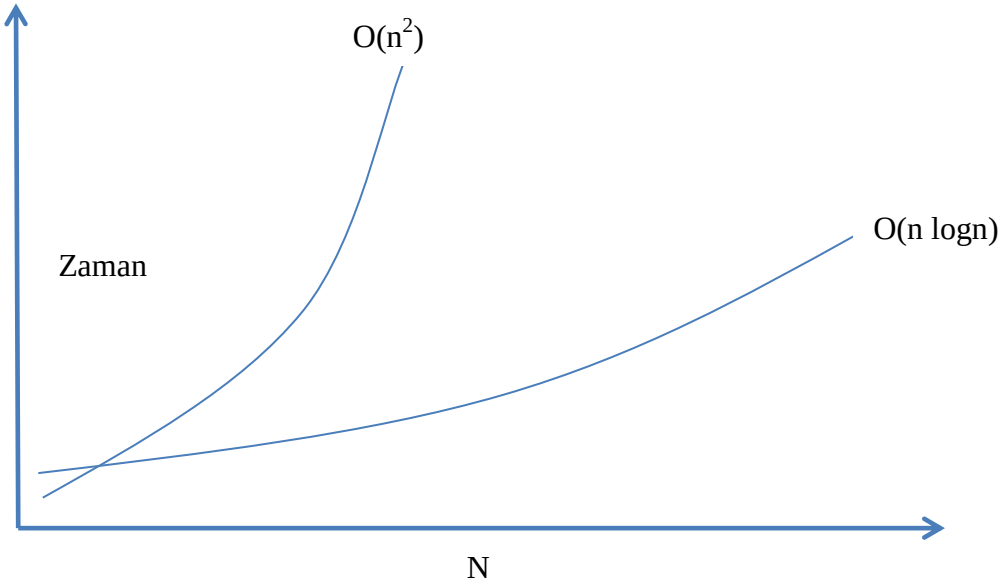
- Kabarcık sıralama iki FOR döngüsü kullanır.
- Dıştaki döngü, işlemin (n-1) defa tekrarlanmasını sağlar.
- İçteki döngü (n-1) kıyaslama yapar.
- n=10000 (döngü sayısı) ve işlem çalışma zamanı "t" varsayılmaktadır.

```
for i:= 0 to 9999 do                                    (n-1)
begin
        for j:= 9999 downto (i+1) do                    (n-1)
        begin
                if A[j] < A [j-1] then
                begin
                        bubble:=A[j];     ];            (t)
                        A[j]:=A[j-1
                        A[j-1]:=bubble;
                end;
        end;
end;
```

Algoritmada, büyük o notasyonu ile çalışma zamanı hesaplanması;

$O(N)=(n-1)*(n-1)*t$
$O(N)=n^2-2n+1+t$
$O(N)=n^2$

Şeklinde gösterilecektir. O notasyonundan farklı olarak, pratikte dizi sıralandıkça çalışma zamanı logaritmik olarak azalacaktır. Ω notasyonu ile algoritmanın çalışma zamanı log(N) olarak ifade edilecektir (Şekil 3.12.).

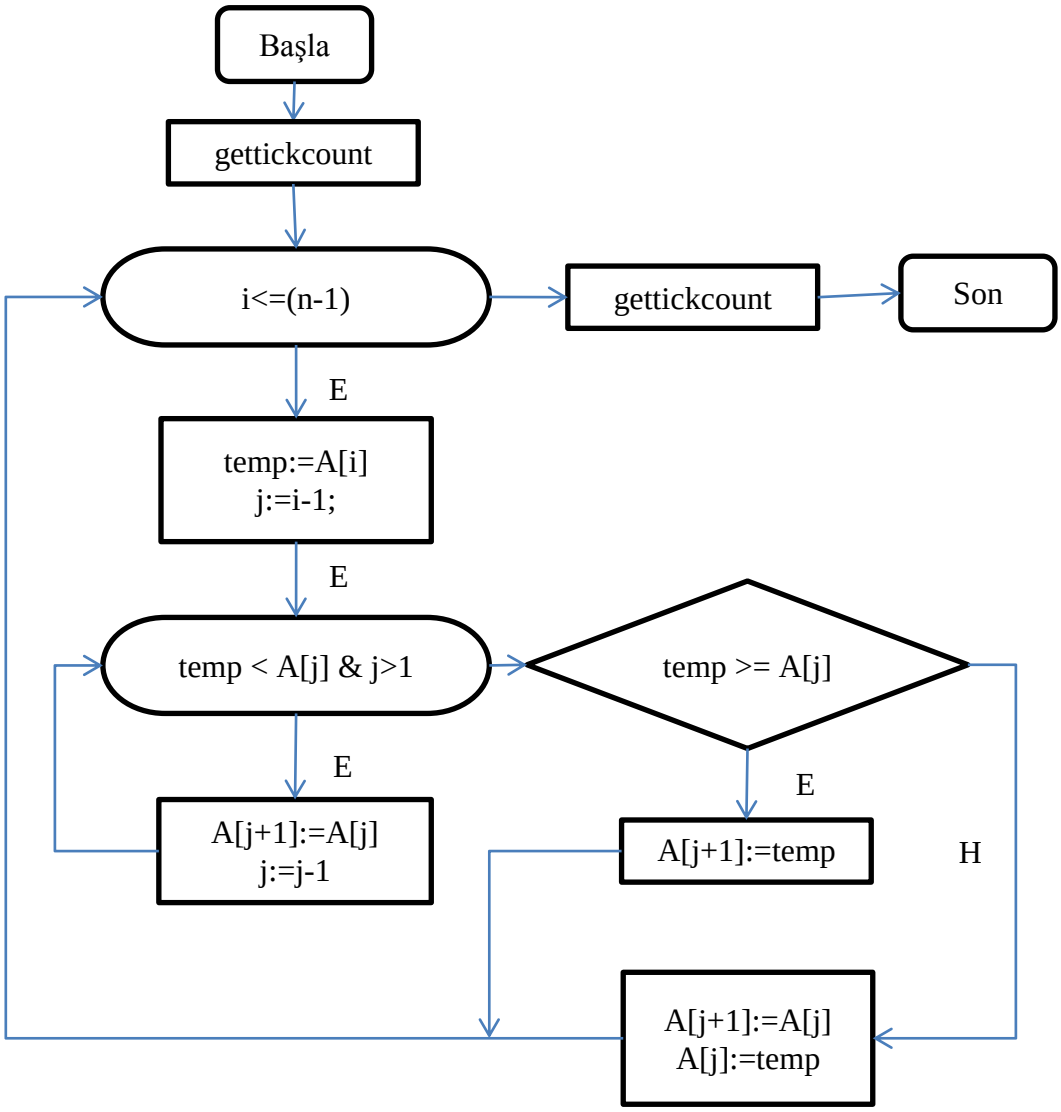


Şekil 3.12. Büyük O ve Ω notasyonu gösterimi.

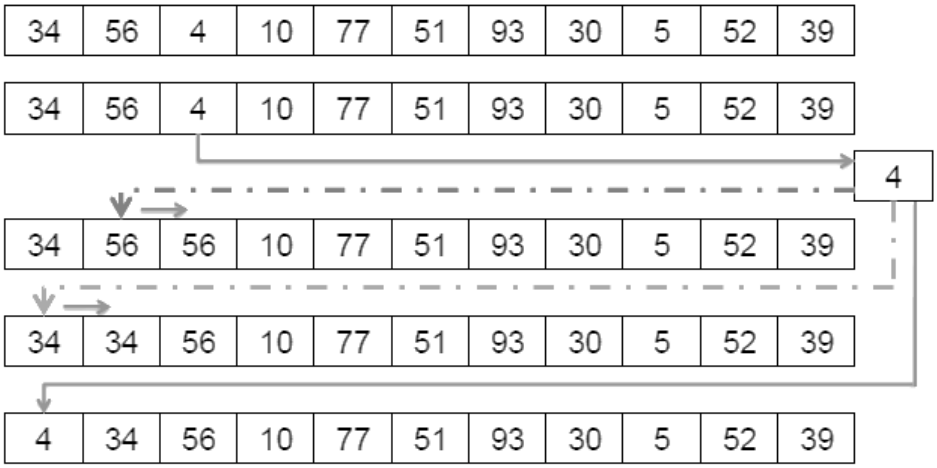
- Insertion Sort (yerleştirmeli sıralama) başka bir basit sıralama algoritmasıdır. Örnek işleyişi şekil 3.14.' te gösterilmiştir. Algoritma adımları;

1. Algoritmanın ilk iterasyonu, dizinin ikinci elemanını alır, ikinci eleman eğer birinci elemandan küçükse, birinciyle yer değiştirir.
2. İkinci iterasyon, daha sonra üçüncü elemana bakar, ilk elemanın durumuna göre doğru yerine yerleştirir, böylece üç eleman da sıralanmış olur.

3. Algoritmanın i. terasyonunda, ilk eleman doğru şekilde sıralanmış olacaktır. Algoritmanın akış şeması şekil 3.13.' te verilmiştir.



Şekil 3.13. Yerleştirmeli sıralama algoritmasının akış şeması.



Şekil 3.14. Yerleştirmeli sıralama algoritmasının işleyişi.

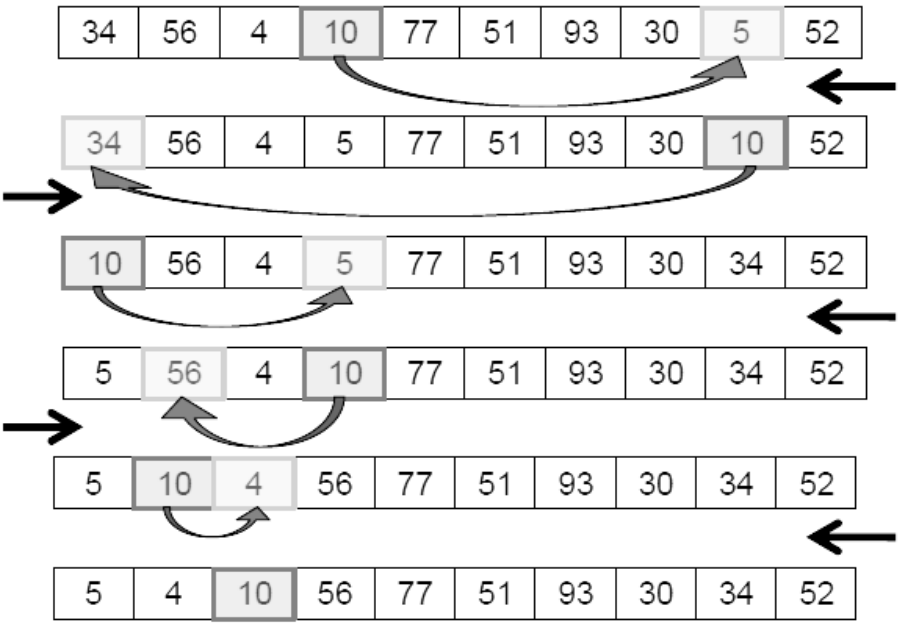
```
for i:= 1 to 9999 do                                          (n-1)
begin
        temp:=A[i];      j:=i-1;                              (t)
        while (temp < A[j]) and (j>1) do                      (n-2)
        begin
                A[j+1]:=A[j];      j:=j-1;
        end;
        if temp >= A[j] then
                A[j+1]:=temp
                Else begin
                        A[j+1]:=A[j];
                        A[j]:=temp;
        end;
end;
```

Yerleştirmeli sıralama algoritması, $O(n^2)$ tipinde bir algoritmadır. Aynı seçimli algoritma gibi iki döngüye sahiptir. for döngüsü, (n-1) kere çalışır ve elemanı sıralanmış elemanlar içindeki doğru yerine yerleştirir. While döngüsü, dizinin kalan elemanları üzerinde aynı işlemini gerçekleştirir. İlk aşamada, (n-1) işlem

gerçekleştirilir, ikinci aşamada (n-2) işlem gerçekleştirilir ve bu şekilde $O(n^2)$ tipinde bir yapı oluşmaktadır.

$O(N)=(n-1)*(n-2)$
$O(N)=n^2-3n+2$
$O(N)=n^2$

- Quicksort (Hızlı sıralama) algoritması, sıralanacak bir sayı dizisini daha küçük iki parçaya ayırıp oluşan bu küçük parçaların kendi içinde sıralanması mantığıyla çalışır. İlk önce sayı dizisinden herhangi bir sayı pivot eleman olarak seçilir. Sayı dizisi, pivot sayıdan küçük olan tüm sayılar pivotun önüne, pivottan büyük olan tüm sayılar pivotun arkasına gelecek biçimde tekrar düzenlenir. Bu bölümlendirme işleminden sonra eleman sıralanmış son dizide olması gerektiği yere yerleştirilir. Pivotun sol ve sağında olmak üzere oluşan iki ayrı küçük sayı dizisi üzerinde, hızlı sıralama algoritması yinelemeli olarak yeniden çağrılır. Algoritma; eleman sayısı sıfır olan bir alt diziye ulaştığında bu dizi sıralanmıştır. Algoritmanın örnek işleyişi şekil 3.15.' de gösterilmektedir.



Şekil 3.15. Hızlı sıralama algoritmasının örnek işleyişi.

```
Lo := iLo;
Hi := iHi;                                          (t)
Pivot := A[(Lo + Hi) div 2];
repeat
        while A[Lo] < Pivot do Inc(Lo) ;
        while A[Hi] > Pivot do Dec(Hi) ;            (?)
        if Lo <= Hi then
        begin
                T := A[Lo];
                A[Lo] := A[Hi];
                A[Hi] := T;                         (?)
                Inc(Lo) ;
                Dec(Hi) ;
        end;
        until Lo > Hi;
if Hi > iLo then QuickSort(A, iLo, Hi) ;
if Lo < iHi then QuickSort(A, Lo, iHi) ;
```

Hızlı sıralama algoritması n adet sayıyı, ortalama veya en iyi durumda O(n log n) karmaşıklığıyla, en kötü durumda ise $O(n^2)$ karmaşıklığıyla sıralar. $O(n^2)$ olan durumlarda, hızlı sıralama algoritması, diğer basit işleyişe sahip sıralama algoritmalarından daha hızlı değildir.

3.2.2.6. İmaj Zip Testi

Bu testte, text zip testi algoritmasının aynısı çalıştırılmaktadır. İmaj zip algoritmasının farkı, bir metin (.txt) dosyası değil, matris format bir resim dosyası (.bmp) sıkıştırılmaktadır.

İmaj zip algoritması ile 8 bitlik bir imaj dosyası, gerçek boyutunun %10‘ una kadar küçültülebilmektedir. Uygulamada kullanılan resim dosyası 8 bit renk derinliğine sahip, 12 MB boyutunda bmp uzantılı bir dosyadır.

İmaj zip testi, aşağıdaki kod satırını kullanmaktadır;

gecensure:=gettickcount;
lboFilesToZip.Items.Add('c:\hbm\PCBench\Temp\image.bmp'); // Ziple
ZipFiles('c:\hbm\PCBench\Temp\image.bmp', lboFilesToZip.Items);// Zip'i kaydet
gecensure:=(gettickcount - gecensure);

3.2.2.7. İmaj Unzip Testi

İmaj zip testi algoritması ile oluşturulan image.zip sıkıştırılmış dosyasınınunzip32.dll kütüphane dosyası fonksiyonları kullanılarak çözülmesi ve diske kaydedilmesi işlemlerini yapan programdır. Aşağıdaki kod satırı bu işlemi göstermektedir;

edtFileToUnzip.Text := 'c:\hbm\PCBench\Temp\text.zip';
edtUnzipToDir.Text := 'c:\hbm\PCBench\Temp\';

3.2.2.8. BMP den JPG ye Çevrim Testi

JPEG standardında görüntü saklayan dosya biçimi de çoğunluk tarafından JPEG olarak adlandırılır. Bu dosyalar genellikle *.jpg*, *.jpe* ya da *.jfif* uzantılıdır, ancak çoğunlukla *.jpg* uzantısı kullanılmaktadır. (Patel vd., 2009) JPEG; dijital kameralarda ve diğer fotoğrafik görüntü yakalama cihazları tarafından kullanılan en yaygın görüntü biçimidir. Her renk bileşeni, 8x8 bloklar halinde ayrık kosinüs dönüşümü ile dönüştürülür, bu sayede resmin enerjisi az sayıda dönüşüm uzayındaki pikselde yoğunlaştırılır. Aşağıdaki (3.1) formül, bmp formatından jpg'ye çevrim yapılırken kullanılan örnek bir matematiksel işlemi göstermektedir.

$$X_{k_1,k_2} = \sum_{n1=0}^{N_1-1} \sum_{n2=0}^{N_2-1} x_{n_1,n_2} \cos\left[\frac{\pi}{N_1}\left(n_1+\frac{1}{2}\right)k_1\right] \cos\left[\frac{\pi}{N_2}\left(n_2+\frac{1}{2}\right)k_2\right].$$

(3.1)

(http://tr.wikipedia.org/wiki/JPEG)

Uygulamada 8 bit renk derinliğinde 1670x7495 çözünürlükte 11,9MB büyüklüğünde, image.bmp dosyası, JPG' e çevirme algoritması kullanılarak 24 bit renk derinliğinde, aynı çözünürlükte, 5,69MB imagetemporary.jpg dosyasına dönüştürülmekte ve bu dönüşüm işleminin başlangıcından bitişine kadar olan süre ölçülmektedir. Aşağıdaki kod satırı, uygulamada gerçekleştirilen dönüşüm işlemini yapmaktadır;

```
Bitmap := TBitmap.Create;
try
Bitmap.LoadFromFile(BMPpic) ;
JpegImg := TJpegImage.Create;
try
JpegImg.Assign(Bitmap) ;
JpegImg.SaveToFile(JPGpic) ;
Result:=True;
finally
JpegImg.Free
end;
finally
Bitmap.Free
end;
end;
```

3.2.3. CPU Mantıksal Test Algoritmaları ve Uygulanması

CPU Mantıksal test aşamasında 5 farklı benchmark algoritması uygulanmıştır. Her algoritma tamamlandıktan sonra ürettiği sayısal skoru bir log dosyasına kaydetmekte ve ana program modülü, bu log dosyalarından skorları okumaktadır. Mantıksal testlerin sonucunda Şekil 3.16.' daki gibi sonuçlar gösterilmektedir.

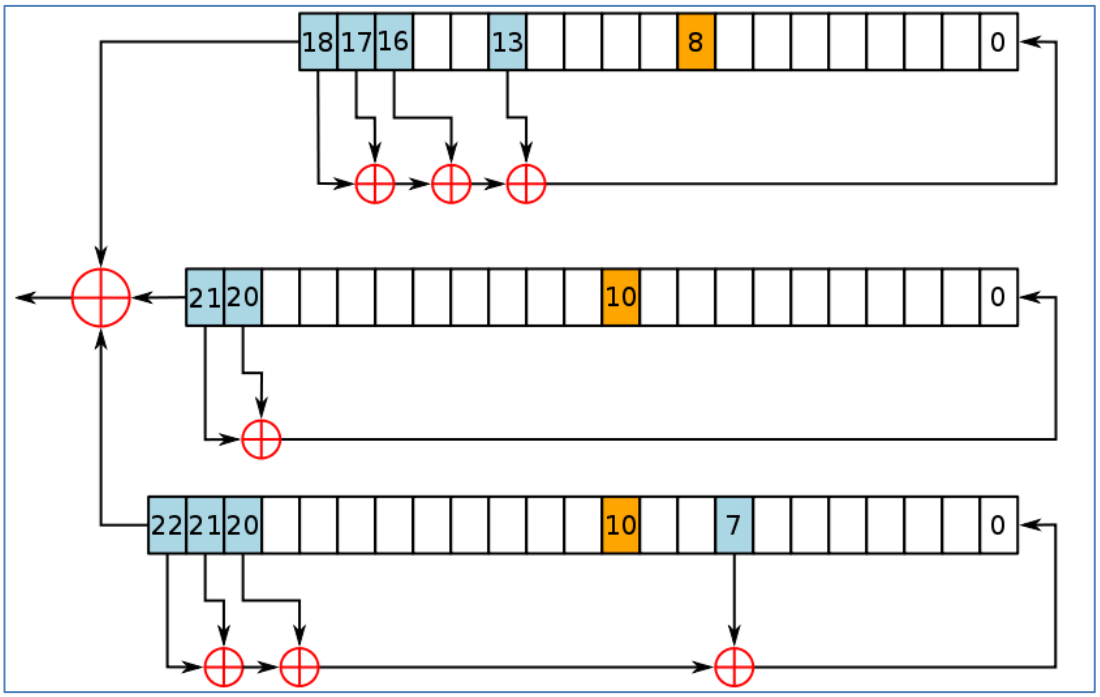
TEST	SKOR	SÜRE	BELLEK	ORTALAMA SKOR
Steam Cipher Kripto	5006	99,9 ms	6968 kb	4108
İmaj sharpen	5282	142 ms	13188 kb	
Gaussian Blur	3862	129,5 ms	5836 kb	
Mandelbrot	3010	996,8 ms	5888 kb	
BMP Enkript	3379	29,6 ms	12292 kb	

Şekil 3.16. Programın, SONY VAIO SR29VN notebook bilgisayarda mantıksal test sonucu gösterimi. (Intel P8600 2Core CPU, 4GB RAM, ATI 3470 GPU)

3.2.3.1. Stream Cipher Kriptoloji Testi

Kriptografide, stream cipher; birkaç basamak şifreleyicinin tek basamakta birleştiği simetrik anahtar şifreleyicidir. Stream cipher' da, her metin, karşılık gelen keystream ile aynı anda şifrelenir. Rasgele keystream, kayan yazmaçlar kullanılarak hızlı bir şekilde üretilir.

Stream cipher, block cipher' lardan simetrik şifrelemeye farklı bir yaklaşım sunar. Block cipher' lar; büyük bloklar üzerinde değişmeyen çevrim yaparlar (Şekil 3.17.). Bu ayrım her zaman net değildir; bazı işlem modlarında, bir blok şifreleme basit bir stream cipher gibi olarak etkili bir şekilde kullanılmaktadır. Stream cipher' lar, block cipher' lara göre daha az sistem kaynağı ile daha hızlı çalışabilir. Stream cipher' lar yanlış kullanıldıklarında sistemde ciddi güvenlik açıkları oluşabilir (http://en.wikipedia.org/wiki/Stream_cipher).



Şekil 3.17. Keystream üretim işlemi A5/1,Cep telefonu görüşmesini şifrelemek için kullanılan bir LFSR tabanlı stream cipher (http://en.wikipedia.org/wiki/Stream_cipher).

Uygulamada; Stream Cipher algoritması kullanarak veri şifreleme ve çözme adımları;

1. Şifreleme anahtarı oluşturma;

```
// şifreleme anahtarı rasgele 16 rakam olarak üretilmektedir
// hepsi rakam olmak zorunda değil, tüm karakterler kullanılabilir
Randomize;
anahtar:=inttostr(random(100000000))+inttostr(random(100000000));
```

2. ….\metin.txt dosyasındaki veriyi anahtarla şifreleyerek C:\kripto\kriptolu.txt dosyasına yazmak;

```
// Dosyanın harddisk' teki yeri gösterilmektedir
assignfile(Dosyadanoku,'C:\hbm\PCBench\Temp\metin.txt');
```

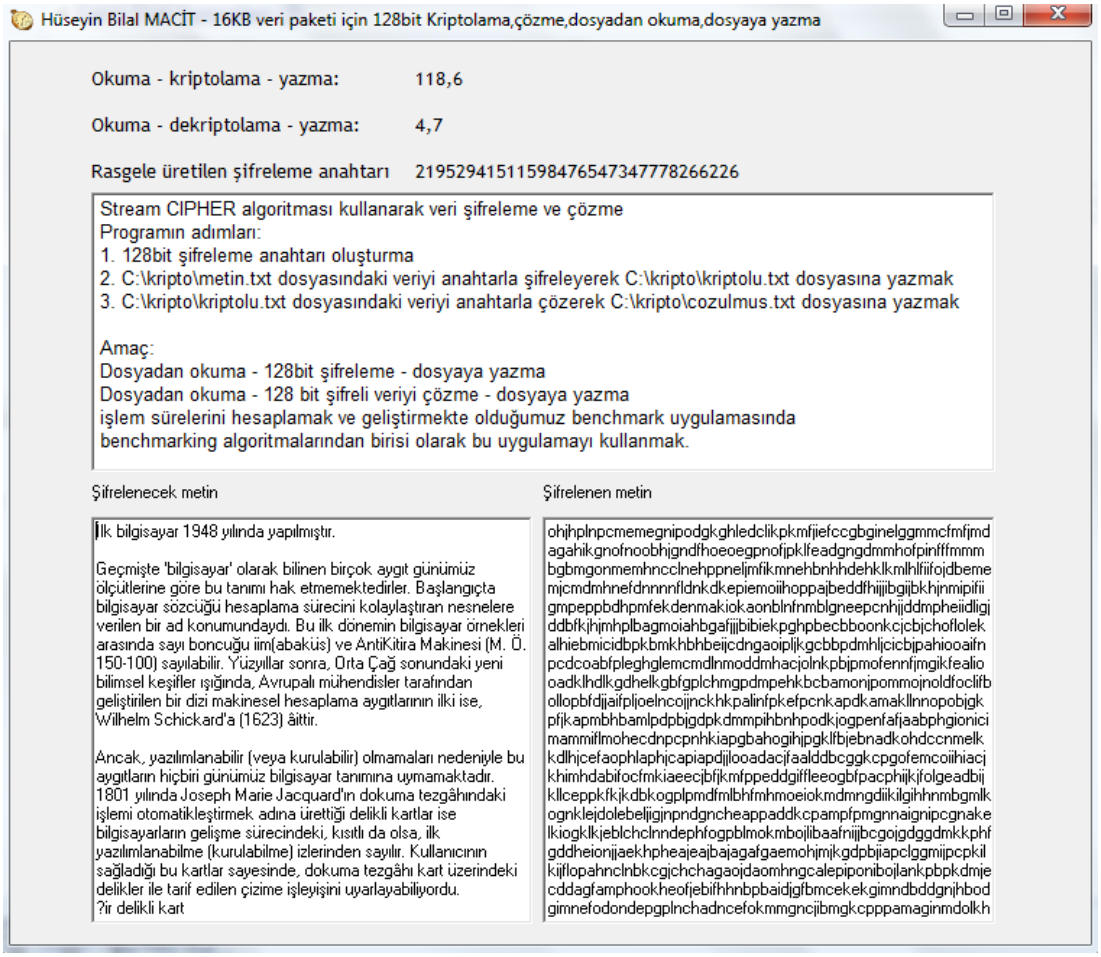
// reset komutu ile dosya, okuma modunda açılmaktadır
reset(Dosyadanoku);
// Kriptolama işleminin yapılması
crypt(Bufkey, buf, Size);
FreeMem(buf);
FreeMem(Bufkey);
Memo2.SetTextBuf(Cry);
// kriptolu string, txt dosyasına yazmak üzere hazırlanır ve dosyaya yazılır.
kriptolu:= cry;
// Dosyanın harddiskteki yerininin gösterimi
assignfile(Dosya,'C:\hbm\pcbench\temp\kriptolu.txt');
// Rewrite komutu ile dosyanın yazma modunda açılması
Rewrite(Dosya);
// kriptolanmış metin, kriptolu stringinden kriptolu.txt dosyasına yazılmaktadır
Write(Dosya,kriptolu);
// dosyanın bellekten silinmesi
closefile(Dosya);

3. C:\kripto\kriptolu.txt dosyasındaki veriyi anahtarla çözerek C:\kripto\cozulmus.txt dosyasına yazmak;

Dosyadan okuma fonksiyonu;
decrypt(Bufkey, buf, Size);
FreeMem(buf);
FreeMem(Bufkey);
Memo1.SetTextBuf(Cry);
cozulmus:=cry;
Dosyaya yazma fonksiyonu

Dosyadan okuduktan sonra ve dosyaya yazmadan hemen önce, kriptolama ve çözme işlemleri yapılmaktadır. “gettickcount” komutu ile sadece şifreleme aşaması için süre okuması yapılarak, bilgisayarın şifreleme performansı ölçülmekte ve loglanmaktadır.

Stream cipher kullanırken en fazla 128 bit şifreleme anahtarı kullanılabilmektedir. Şifreleme işleminde tampon bellek kullandığı için, 128 bitten daha büyük anahtar kullanıldığında "Stack overflow" hatası alınabilmektedir. Şifreleme işleminin yapıldığı ekran, süre ölçüm sonuçları, şifrelenmiş ve orijinal metin dosyası şekil 3.18.' de gösterildiği gibidir.



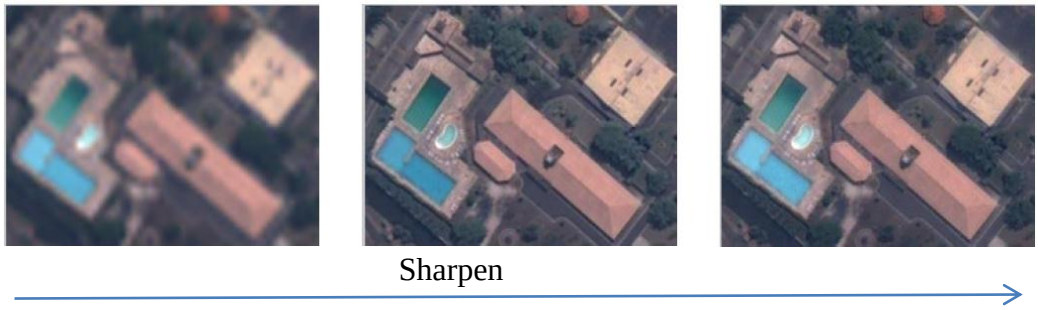
Şekil 3.18. Stream cipher işlemini yapan programın ekran görüntüsü.

3.2.3.2. İmaj Sharpen Testi

Sharpen; kelime anlamı olarak keskinleştirmek, bilemek demektir. Bir resmi sharpen işlemine tabi tutmak, o resmi keskinleştirmek anlamına gelmektedir. Şekil 3.19.' deki örnekte görüldüğü üzere, aynı fotoğrafın, sharpen efekti uygulanmış ve orijinal

hali yan yana koyulduğunda, sharpen efekti uygulanmış hali daha detaylı görülebilmektedir. (Hogan, 2003)

Adobe photoshop ve benzeri resim düzenleme yazılımlarında sıkça kullanılan sharpen efekti, alttaki şekilde görüldüğü gibi uydudan çekilmiş fotoğraflarını keskinleştirme amacı ile de çok fazla kullanılmaktadır.



Şekil 3.19. Bir resme sharpen efektinin art arda uygulanması. (Rahmanivd., 2008)

Uygulamada her piksel, sharpen işlemi yapılırken etrafındaki piksellerle ele alınır. Aşağıdaki filter dizisi, bir pikselin etrafındaki piksellerin sharpen uygulanmış halini ifade etmek için kullanılır;

```
Filter: array[0..8] of integer; // matris 3 * 3 pixel
Red, Green, Blue, NewR, NewG, NewB, I,
PosX, PosY, mX, mY, dX, dY, Diviseur : integer;
……..
//piksel pozisyonu
mY := PosY + dY;
mX := PosX + dX;
…….
Red := TabScanlineBmp[mY,mX].RGBTRed;
Green := TabScanlineBmp[mY,mX].RGBTGreen;
Blue := TabScanlineBmp[mY,mX].RGBTBlue;
……
//Yeni renk değerlerini oluştur:
```

*NewR := NewR + Red * Filter[I];*

*NewG := NewG + Green * Filter[I];*

*NewB := NewB + Blue * Filter[I];*

Uygulamanın sharpen benchmark işleminde, Test.jpg dosyası 3 kez farklı çarpanlarla sharpen yapılmaktadır. Şekil 3.20.' de, uygulamanın sharpen sonuçları gösterilmiştir:

Çarpan: 50 Çarpan: -50 Çarpan: -100

Şekil 3.20. Farklı çarpan değerleri ile uygulamanın verdiği sonuçların gösterimi

Aşağıdaki kod satırı, uygulamada sharpen efektini yapmaktadır;

```
Bmp:= TBitmap.Create;
try
Bmp.Assign(OrigBmp);
Bmp.PixelFormat:= pf24bit;
BmpAccentuation(Bmp, -50); // -50; çarpan değeridir.
finally
bmp.Free;
end;
```

3.2.3.3. Gaussian Blur Testi

Gaussian Blur; diğer adıyla Gaussian smoothing, birçok benhmark yazılımında kullanılan karmaşık bir matematiksel algoritmadır. Genellikle resim işleme yazılımlarında, resimdeki gürültüyü veya netliği azaltmak için kullanılır. Gaussian

Blur, bir görüntüdeki her piksele bir Gauss fonksiyonu (bu fonksiyon, istatistikte normal dağılımı ifade eder.) uygulamaktır.

Bir boyutlu bir Gauss denklemi;

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}} \tag{3.2}$$

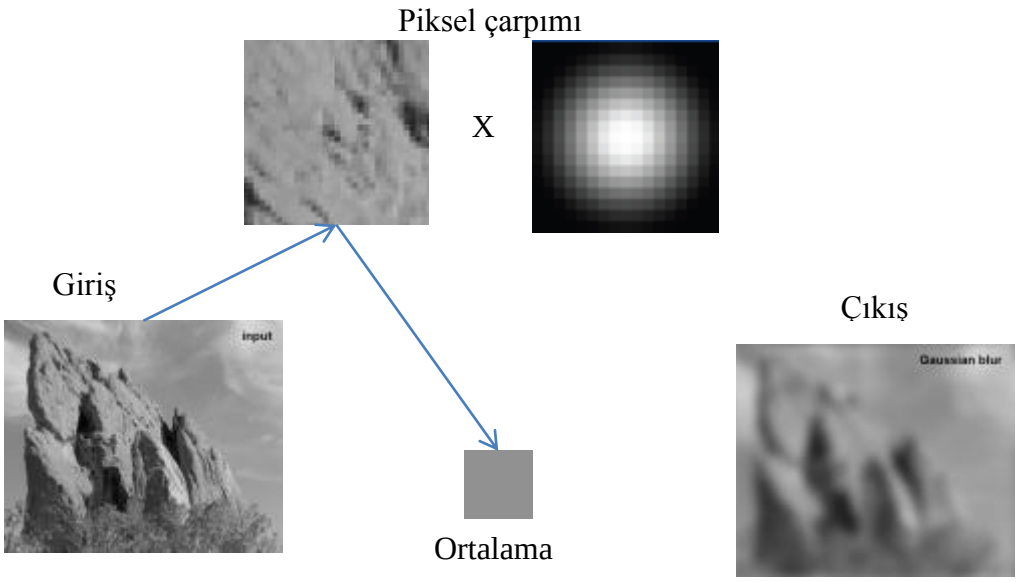
İki boyutlu bir Gauss denklemine eşittir.

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{3.3}$$

3.2 ve 3.3 denklemlerinde;

x: yatay eksende kökenli mesafedir.
y: dikey eksende kökenli mesafedir.
σ: gauss dağılımının standart sapmasıdır.

İki boyutlu uygulandığı zaman, bu formül, sanal bir yüzey oluşturur (Şekil 3.21.). Bu yüzey Gauss dağılımı ile merkezden çıkan dairelerden oluşmaktadır. Bu dağılım değerleri, orijinal görüntü uygulanan bir evrişim matrisi oluşturmak için kullanılmaktadır. Her pikselin değeri, etrafındaki piksellerin ağırlığına göre ayarlanır. Orijinal piksel değeri, en yüksek Gauss ağırlığına sahiptir ve çevredeki piksellerin orijinale uzaklığı arttıkça ağırlığı azalır (http://en.wikipedia.org/wiki/Gaussian_blur).



Şekil 3.21. Gaussian Blur, piksel bazında ifadesi (Paris, 2010)

Uygulamada Gaussian Blur işlemi, aşağıdaki kod satırı ile gerçekleştirilmiştir;

```
//bitmap yerini kaydet
for Row:= 0 to Bitmap.Height - 1 do
Rows[Row]:= Bitmap.Scanline[Row];

//her satıra blur uygula
P:= AllocMem(Bitmap.Width*SizeOf(TRGBTriple));
for Row:= 0 to Bitmap.Height - 1 do
BlurRow(Slice(Rows[Row]^, Bitmap.Width), K, P);

//her sütuna blur uygula
ReAllocMem(P, Bitmap.Height*SizeOf(TRGBTriple));
for Col:= 0 to Bitmap.Width - 1 do

begin
//- bir sütunu TROW a okut
for Row:= 0 to Bitmap.Height - 1 do
```

```
ACol[Row]:= Rows[Row][Col];
BlurRow(Slice(ACol^, Bitmap.Height), K, P);
// tekrar bu satırı veriye yerleştir
for Row:= 0 to Bitmap.Height - 1 do
Rows[Row][Col]:= ACol[Row];
end;
```

Yukarıdaki kod satırı kullanılarak uygulamada elde edilen Gaussian Blur efekti uygulanmış görüntü, şekil 3.22.' de gösterildiği gibidir.

Orijinal Nesne Gaussian Blur

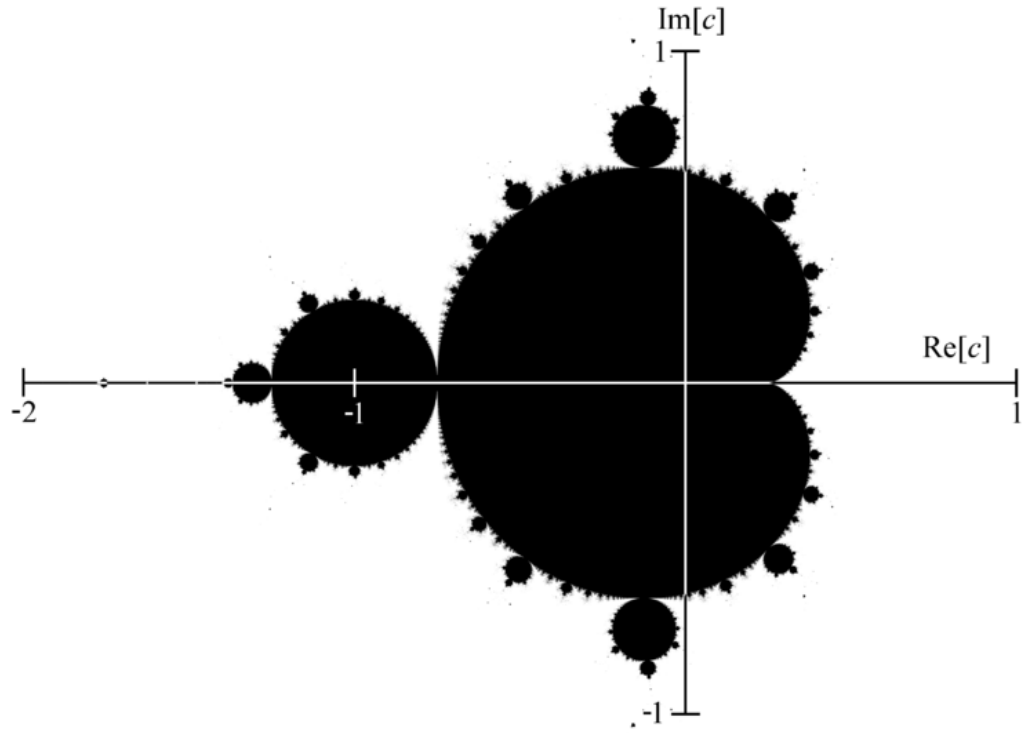
Şekil 3.22. Geliştirilen uygulamanın Gaussian Blur efekti uygulaması

3.2.3.4. Mandelbrot Set Testi

Mandelbrot seti, iki boyutlu bir fraktalın oluşturduğu noktalar kümesidir. Fraktal terimi parçalanmış yada kırılmış anlamına gelen Latince "fractus" sözcüğünden türetilmiştir. İlk olarak 1975' te Polonya asıllı matematikçi Beneoit B. Mandelbrot tarafından ortaya atılmıştır.

Kendi kendini tekrar eden ama sonsuza kadar küçülen şekilleri, kendine benzer bir cisimde cismi oluşturan parçalar ya da bileşenler cismin bütününü inceler. Düzensiz ayrıntılar ya da desenler giderek küçülen ölçeklerde yinelenir ve tümüyle soyut

nesnelerde sonsuza kadar sürebilir; tam tersi de her parçanın her bir parçası büyütüldüğünde, yine cismin bütününe benzemesi olayıdır. Fraktal ne kadar büyütülürse büyütülsün, bakış açısı nasıl değiştirilirse değiştirilsin, hep aynı görüntü elde edilir. Eğrelti otu, kar taneleri, brokoli ve galaksilerin yapısı fraktallara örnek olarak verilebilir.

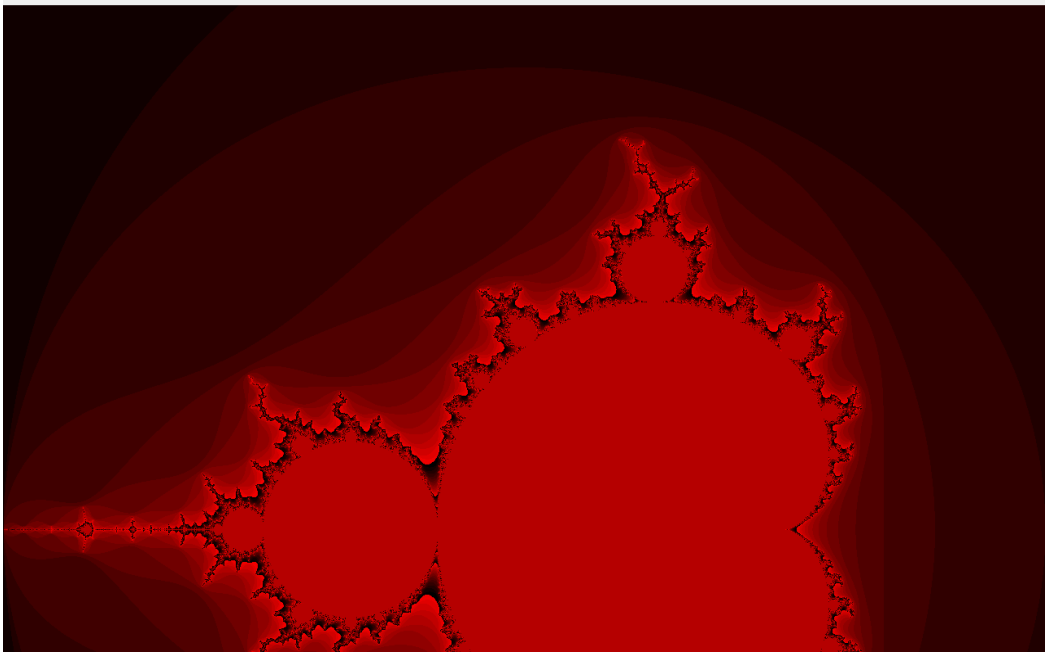


Şekil 3.23. Mandelbrot tasviri. (Im[c]; c nin sanal ve Re[c] c' nin gerçel parçasıdır.)

Benchmark yazılımlarının birçoğunda fraktal çizimi kullanılmaktadır. Benchmark uygulamalarında kullanılan en yaygın fraktal çizimi şekil 3.23.' dekidir. Geliştirilen uygulamada çizilen mandelbrot da buna uygundur (Şekil 3.24.). Manldelbrot performansı, makinenin floating point performansını gösterdiği için işlemci FPU / mantıksal test bölümünde gösterilmektedir. Benchmark yazılımlarında mandelbrot performansı, Lua performansı olarak da gösterilmektedir. (http://en.wikipedia.org/wiki/Mandelbrot_set)

Uygulamada mandelbrot çizdiren kod satırı ve çizim sonucu aşağıda verilmiştir;

```
c2 := bu;
for i := 10 to X2 do
begin
c1 := au;
for j := 0 to Y2 do
begin
z1 := 0;
z2 := 0;
Count := 0;
{Mandelbrot kümesi yineleme sayısı
if |z| >=2 ise z mandelbrot üyesi olamaz}
while (((z1 * z1 + z2 * z2 < 4) and (Count <= 90))) do
begin
tmp := z1;
z1 := z1 * z1 - z2 * z2 + c1;
z2 := 2 * tmp * z2 + c2;
Inc(Count);
end;
// renk paleti TColor(n*count mod t)
{$IFDEF LINUX}
ACanvas.Pen.Color := (16 * Count mod 255);
ACanvas.DrawPoint(j, i);
{$ELSE}
ACanvas.Pixels[j, i] := (16 * Count mod 255);
{$ENDIF}
c1 := c1 + X;
end;
c2 := c2 + Y;
end;
```



Şekil 3.24. Geliştirilen uygulamanın çizdiği mandelbrot.

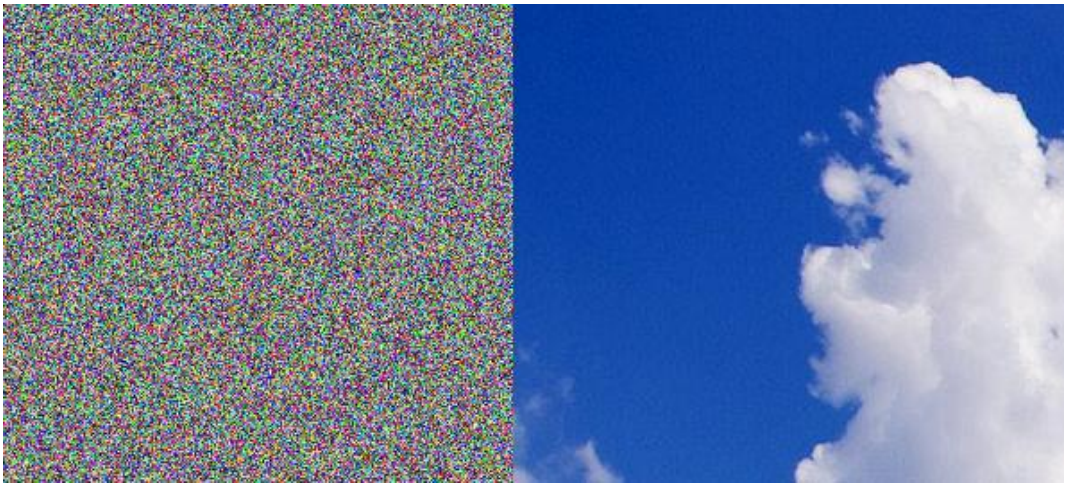
3.2.3.5. BMP Enkript Testi

Encription; Türkçe karşılığı olarak şifreleme; bilgi güvenliğini arttırmak amacıyla, bilgiyi şifreleyerek başkalarının anlayamayacağı bir biçime dönüştürme işlemidir.

Enkript algoritmaları, işlemcinin kayan nokta ve integer işlem performansını ölçmede son derece etkilidir. Bu nedenle birçok benchmark yazılımında kullanılmaktadır. Şekil 3.25.' te, sol taraf algoritma ile şifrelenmiş, sağ taraf da orijinal bitmap' ı göstermektedir. Geliştirilen uygulamada, aşağıdaki kod satırı uygulanarak, bitmap dosyası 10 defa şifrelenmektedir.

```
try
BytesPorScan:=Abs ( Integer(BMP.ScanLine[1])-
Integer(BMP.ScanLine[0]));
except
raise exception.create('Hata');
end;
```

```
RandSeed:=Clave;
for h:=0 to BMP.Height-1 do
begin
P:=BMP.ScanLine[h];
for w:=0 to BytesPorScan-1 do
P^[w]:=P^[w] xor Random(256);
end;
```



Şekil 3.25. Uygulamada şifrelenmiş ve orijinal resim.

3.2.4. RAM Testi Algoritmaları ve Uygulanması

RAM ve işlemci hızları arasında zamanla oluşmaya başlayan uçurum, programcıları belleği daha etkin kullanan uygulamalar geliştirmeye yönlendirmiştir. Bellekteki kısıtlı alanın etkin kullanımı için, verilerin belleğe diziliminden, kullanılmayanların hemen bellekten çıkartılmasına kadar birçok işlemi hesaplamak gerekmektedir.

Bellek miktarı yetersiz ya da bellek performansı düşük bir sistem, işlemcinin performansının ciddi anlamda düşük çalışmasına neden olacaktır. Geliştirilen uygulamada, bellek performansının ölçümü aşamasında, tek bir algoritma, farklı veri sayıları kullanarak ardışık ve rasgele olarak belleğe yazma ve silme işlemleri yapmaktadır. Her yazma işlemi bittikten sonra, algoritmayı çalıştıran modül yazma

süresini log dosyalarına kaydetmekte ve ana program modülü, bu log dosyalarından skorları okumaktadır. RAM testleri sonucunda Şekil 3.26.' deki sonuçlar gösterilmektedir.

TEST	SKOR	SÜRE	BELLEK	ORTALAMA SKOR
A 10byte 1000blok	4007	624 ms	9 kb	4204
R 10byte 1000blok	4588	109 ms	11 kb	
A 100byte 1000blok	4588	109 ms	51 kb	
R 100byte 1000blok	4000	125 ms	52 kb	
A 1000byte 1000blok	4361	172 ms	499 kb	
R 1000byte 1000blok	4808	156 ms	512 kb	
A 4096byte 1000blok	3572	140 ms	2074 kb	
R 4096byte 1000blok	3206	156 ms	2043 kb	
A 10000byte 1000blok	6000	250 ms	4934 kb	
R 10000byte 1000blok	5661	265 ms	4873 kb	
A 4097-10000byte 100blok	4808	312 ms	6911 kb	
R 4097-10000byte 100blok	3847	390 ms	6928 kb	
A 100000byte 100blok	3290	608 ms	5331 kb	
R 100000byte 100blok	3373	593 ms	5044 kb	
A 100000byte 30blok	3368	297 ms	12153 kb	
R 100000byte 30blok	3774	265 ms	14128 kb	

Şekil 3.26. Programın, SONY VAIO SR29VN notebook bilgisayarda RAM test sonucu. (Intel P8600 2Core CPU, 4GB RAM, ATI 3470 GPU)

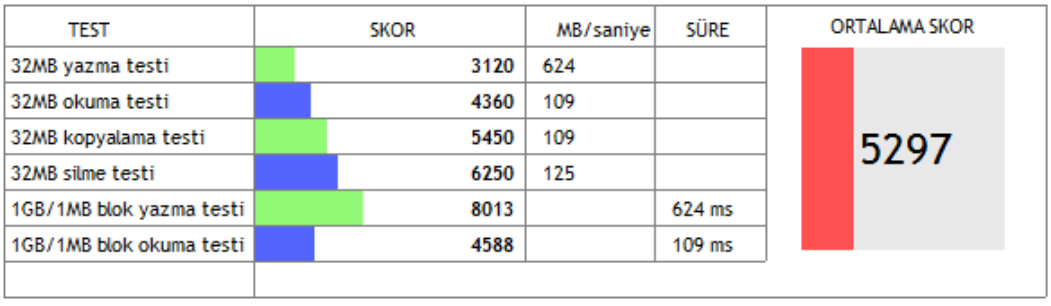
Aşağıdaki kod satırı, uygulamada, RAM Allocation işlemini gerçekleştirmektedir;

```
if IsStdManager then
SetMemoryManager(StdManager);
FillChar(A,SizeOf(A),0);
D := GetTickCount;
if Sequential then // Yazma işlemi ardışık mı ?
for I := 1 to 1000 do
for J := 1 to 1000 do
begin
FreeMem(A[J]); // Belleğe son yazılan silinir
GetMem(A[J],Random(10000)+1); //10,000 byte belleğe yazılır
end
```

```
else
for I := 1 to 1000 do
for J := 1 to 1000 do
begin
K := Random(1000)+1;
FreeMem(A[K]);// Belleğe son yazılan silinir
GetMem(A[K],Random(10000)+1);//10,000 byte belleğe yazılır
end;
D := GetTickCount-D;
if IsStdManager then
begin
StdAllocSize := AllocMemSize;
SetMemoryManager(QMemManager);
end;
```

3.2.5. Harddisk Testi Algoritmaları ve Uygulanması

Harddisk; temel işlevleri yazma ve okuma olan sistem bileşenidir. Dolayısıyla yazma ve okuma işlemlerinin ayrı ayrı test edilmesi gerekmektedir. Geliştirilen uygulama, harddiske puanlama yapabilmek için, Şekil 3.27.' da sonuçları gösterilen 6 faklı test algoritması uygulamaktadır. Uygulama, ilk 4 testin okuma yazma hızlarını MB/saniye cinsinden hesaplayabilmektedir.



Şekil 3.27. Programın, SONY VAIO SR29VN notebook bilgisayarda HDD test sonucu gösterimi. (Intel P8600 2Core CPU, 4GB RAM, ATI 3470 GPU, Toshiba 320GB HDD)

32MB yazma testi, HDD' de oluşturulan boş bir dosyanın içine 32MB'a kadar veri yazılması hızını ölçmektedir. Aşağıda bu işlem için kullanılan kod satırı bulunmaktadır:

```
sFileName := 'C:\hbm\PCBench\Temp\data128.dat';
FindFirst(sFileName,faAnyFile,srFileSR);
sFileSize := IntToStr(srFileSR.Size);
AssignFile(data128,'C:\hbm\PCBench\Temp\data128.dat');
gecensure:=gettickcount;
while StrToInt(sFileSize)<33554432do // Buradaki 33554432değeri, dosyanın sonlarndırılma boyutudur. Bu dosya 32MB' a ulaşınca sonlandırılacaktır.
begin
FindFirst(sFileName,faAnyFile,srFileSR);
sFileSize := IntToStr(srFileSR.Size);
append(data128);
Write(data128,'01010101010101010101010101010101010101010101010101010101010);
CloseFile(data128);
end;
gecensure:=(gettickcount - gecensure);
gecensuresalise:=gecensure/10;
……..
//MB/saniye oranının hesaplanması
sureoran:=100*128/gecensuresalise/1024;
```

Yukarıdaki kod satırında oluşturulan 32MB dosya ile, Delphi' nin read, copy ve delete fonksiyonları kullanılarak, okuma, kopyalama ve silme işlemleri süreleri hesaplanarak harddisk performansı ölçülmekte ve log dosyalarına skorlar kaydedilmektedir.

Uygulamada, ikinci harddisk testi, bloklar halinde yazma ve okuma testidir. Aşağıdaki kod satırı blok yazma işlemini gerçekleştirmektedir;

```
SetLength(ByteArr, ArrayLen);
AssignFile(f, 'C:\hbm\pcbench\temp\datatest.dat');
Rewrite(f, RecSize);
Randomize;
for i := low(ByteArr) to high(ByteArr) do
ByteArr[i] := Random( high(Byte)); // rasgele Byte ifadeler ile dizi doldurulur
for i := 0 to 1023 do
BlockWrite(f, ByteArr[0], BuffCount);
CloseFile(f);
```

Aşağıdaki kod satırı ile yazılan veri bloklar halinde okunmaktadır;

```
SetLength(ByteArr, ArrayLen);
AssignFile(f, 'C:\hbm\pcbench\temp\datatest.dat');
Reset(f, RecSize);
repeat
BlockRead(f, ByteArr[0], BuffCount, ReadLen); // 1MB lik bloklar halinde oku

until ReadLen < BuffCount;
CloseFile(f);
```

3.2.6. Grafik Testi Algoritmaları ve Uygulanması

Grafik kartının temel amacı, üretilen sonuçları ekran bileşenine aktarmaktır. Ancak son yıllarda, gelişen oyun teknolojisi ile birlikte, grafik kartları kendi işlemcileri ve kendi belleklerine sahip karmaşık bileşenler halini almıştır. Ekran kartları kıyaslanırken birçok farklı kriter baz alınır;

- **Piksel yazma oranı (Pixel write rate):** Saniyede ekrana kaç piksel yazılabildiğini ifade eder.

- **Piksel karışım oranı (Pixel blend rate):** Bir yüzeyin üzerinde, ona bağıntılı ikinci bir yüzey oluşturmadan önce, önceki yüzeydeki piksellerin okunması hızını ifade eder.
- **Saniyede çokgenler (Vertex transform rate):** Ekran çizilen çokgen oranını ifade eder. Bu çokgenler genellikle üçgen olarak kullanılır ve grafik kartı üretici firmalar, ürettikleri kartların saniyede çizebildiği üçgen sayısını performans kriteri olarak kullanıcıya sunar.
- **Yüzey yükleme oranı (Texture upload rate):** Eğer ekrana çizilen yüzey için gerekli bellek miktarı, grafik kartının bellek miktarından daha yüksekse, bu verinin sistem belleğine (RAM) gönderilmesi gerekmektedir. Buradaki veri aktarım hızı, sistemin özelliğine bağlı olarak sabit tanımlanmıştır. Örneğin AGP, AGP 2x, AGP 4x, AGP 8x, PCI-EX 16x terimlerindeki hız çarpanları gibi.
- **Vertex veri aktarım oranı (Vertex data transfer rate):** İşlemci yada bellekteki verilerin ekrana çizilmesi gerekiyorsa, bu verinin grafik kartına gönderilmesi gerekmektedir. Bu yükleme işlemine genel olarak "texture upload" denilmektedir. CPU yada bellekten grafik kartına gönderim hızı, vertex veri aktarım oranı olarak ifade edilmektedir.
- **Program karmaşıklığı (Program complexity):** Yapılmakta olan çizim, ışıklandırma, gölge oluşturma, yumuşatma, keskinleştirme vb. efektler kullanıyorsa, çizim için çok daha farklı matematiksel işlemler gerekecek ve çizim gecikecektir. Buradaki süre ifadesi, program karmaşıklığıdır (http://www.mindcontrol.org/~hplus/graphics/ogl-perf).

Yukarıdaki kriterler, farklı uygulamalarda, farklı algoritmalarda farklı sonuçlar vermektedir. Geliştirilen uygulamada, grafik işlemcinin hem 2D, hem 3D performansı test edilebilmekte ve skorlanmaktadır. Şekil 3.28.' de, test sonucunda oluşturulan örnek bir skor tabelası gösterilmiştir;

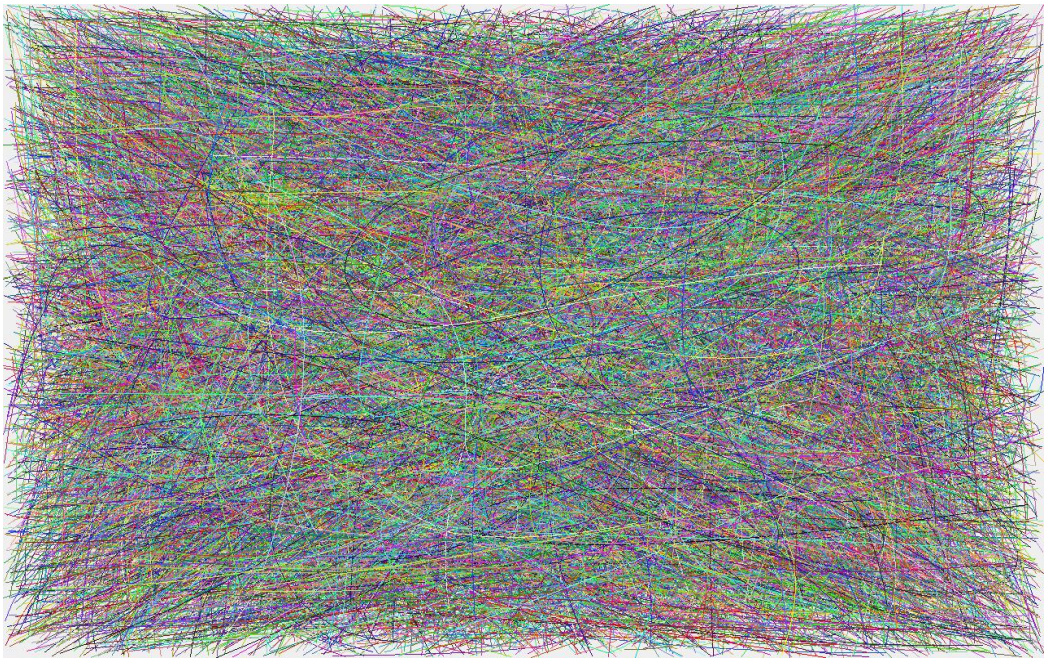
TEST	SKOR	SÜRE	BELLEK	ORTALAMA SKOR
Rasgele Canvas	4855	103 ms	5636 kb	3521
Pikselize	4027	1241,7 ms	5828 kb	
Döşeme	510	98,2 ms	6048 kb	
Screencapture	1087	92 ms	49616 kb	
Negatif imaj	7650	653,6 ms	5740 kb	
OpenGL FPS Test	2180	109 fps		
Boyutlandırma	4333	115,4 ms	7604 kb	

Şekil 3.28. Programın, SONY VAIO SR29VN notebook bilgisayarda grafik test sonucu gösterimi (Intel P8600 2Core CPU, 4GB RAM, ATI Radeon HD3470 GPU 256MB VRAM)

3.2.6.1. Rasgele Canvas Testi

Canvas, ekranın rasgele bölümlerine, eşit uzunlukta belli sayıda eğik çizgi çekilmesidir. Uygulamada, Şekil 3.29.' de görüldüğü gibi çizim yapılmaktadır. Basit ve etkili bir grafik testi algoritmasıdır. Aşağıdaki kod satırı ile uygulamada kullanılmıştır;

```
for j:=1 to 10000 do //10,000 adet çizgi çek
begin
SetLength(pusat, 4); // Çizgilerin uzunluğunu tanımlar
for i := Low(pusat) to High(pusat) do
begin
pusat[i].x := Random(Width);
pusat[i].y := Random(Height);
end;
Canvas.pen.Color:=RGB(Random(255),Random(255),Random(255));
Canvas.PolyBezier(pusat);
end;
```



Şekil 3.29. Uygulamada canvas çizim testinin gösterimi.

3.2.6.2. Pikselize Testi

Ekrana pikseller halinde çizim yaptırma işlemdir. Birçok benchmark yazılımında kullanılan basit ve etkili bir performans ölçüm aracıdır. Uygulamada aşağıdaki kod satırı ile kullanılmıştır;

```
iz:=0;
while iz<=Screen.Height do
begin
inc(sayac);
iz:=iz+100;
r1:=k*(iz+30) mod 255;    r2:=k*(iz+60) mod 255;    r3:=k*(iz+90) mod 255;
for j:=(iz-100) to iz do
for i:=0 to Screen.Width do
begin
Canvas.Pixels[i,j+30]:=RGB(r1,r2,r3);
end;
```

3.2.6.3. Döşeme Testi

Canvas fonksiyonu yardımıyla, bir resim dosyasının, ekranın çeşitli yerlerine tekrar tekrar çizilmesi ile yapılan bu benchmark, çizim başlangıcından bitişine kadarki süreyi ölçerek, bilgisayarın 2D performansını ölçmede etkilidir.

3.2.6.4. Screencapture Testi

Ekrandaki görüntünün, işletim sisteminin hazır fonksiyonları ile belleğe aktarılması işlemidir. Yüzey yükleme oranı (Texture upload rate) ölçümü için etkili bir yöntemdir. Günümüz bilgisayarları grafik bellekleri yüksek kapasitede olduğu için, ölçümü bir defa yapmak, performans kriteri oluşturmakta yeterli değildir. Bu yüzden işlemin defalarca tekrarlanması gerekmektedir. Geliştirilen uygulamada, bu işlem aşağıdaki kod satırının 10 defa gerçekleştirilmesi ile yapılmaktadır;

```
DC := GetDC (GetDesktopWindow);
ABitmap:=TBitmap.Create;
try
ABitmap.Width  := GetDeviceCaps (DC, HORZRES);
ABitmap.Height := GetDeviceCaps (DC, VERTRES);
BitBlt(ABitmap.Canvas.Handle, 0, 0, ABitmap.Width,
ABitmap.Height,DC, 0, 0, SRCCOPY);
finally
ReleaseDC (GetDesktopWindow, DC);
end;
Result:=ABitmap;
```

3.2.6.5. Negatif İmaj Testi

Bir bitmap dosyasının, defalarca negatifini alarak ekranda gösterimi işlemidir. Bitmap dosyasının negatifini almak, o dosyanın her pikselinin RGB skalasındaki

Red, Green, Blue değerlerini 255' ten çıkarmaktır. Aşağıda uygulamada kullanılmakta olan örnek negatif imaj algoritmasının kodları verilmiştir;

```
For X := 0 To Image1.Canvas.Width - 1 Do
For Y := 0 To Image1.Canvas.Height - 1 Do
Image1.Canvas.Pixels[X, Y] := RgbToColor
(255 – GetRValue(Image1.Canvas.Pixels[X, Y]),
255 - GetGValue(Image1.Canvas.Pixels[X, Y]),
255 - GetBValue(Image1.Canvas.Pixels[X, Y]));
```

3.2.6.6. OPENGL FPS Testi

Open GL (Open Graphics Library), Türkçe anlamıyla açık grafik kütüphanesi anlamına gelmektedir. Gelişmiş donanım desteği kullanarak üç boyutlu grafikler çizmek için kullanılan, ücretsiz bir grafik uygulama geliştirme arabirimidir. Windows, Linux, Mac, Solaris işletim sistemlerinde OpenGL kütüphane dosyaları mevcuttur. Platformdan bağımsız çalıştığı ve ücretsiz olduğu için benchmark uygulama geliştiricileri tarafından sıkça kullanılmaktadır (Pabeda, 2006).

Geliştirilen uygulamada kullanılan 3D testinde amaç, grafik kartının üç boyutlu şekli çizerken saniyede kaç kare ürettiğini gösteren FPS değerini alabilmektir. Uygulama, ekrandaki diğer bileşenlerden etkilenmemek için tam ekran modunda çalışmaktadır. Burada önemli kriter, bir sistemin farklı çözünürlüklerde üreteceği FPS değerinin farklı olduğunu göz ardı etmemek amacıyla, tüm sistemleri aynı çözünürlükte test etmektir. Uygulama, çalıştığı anda,

```
glCreateWnd(800, 600, true, 32);
```

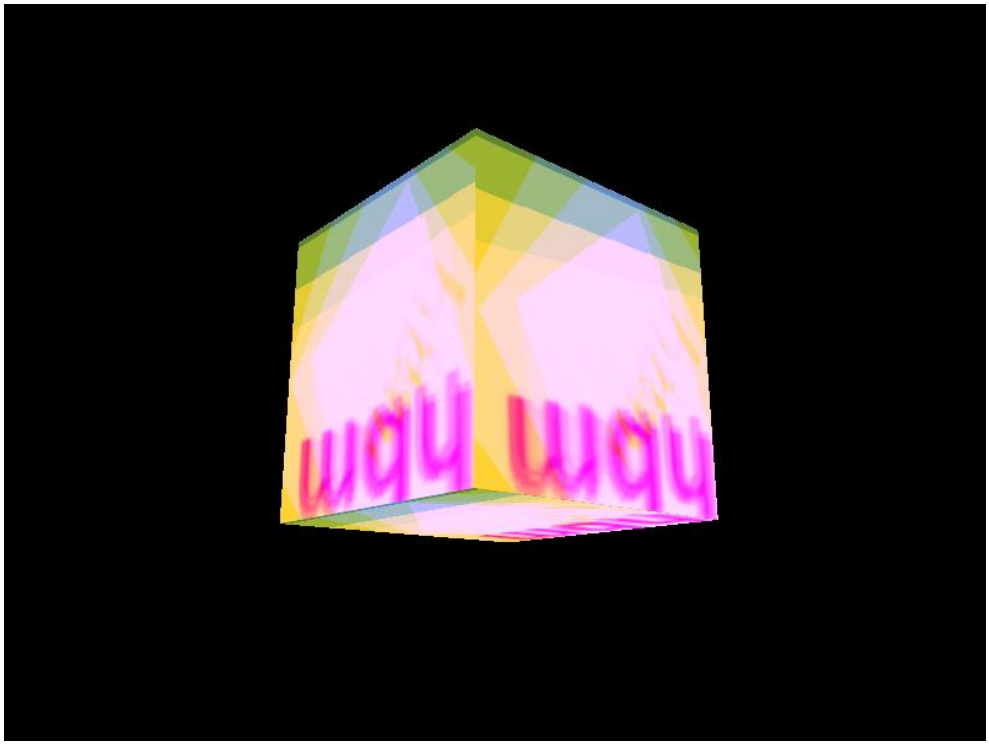
komutu ile ekran çözünürlüğünü 800x600, ve renk sayısını 32 bit olarak ayarlamaktadır.

```
Inc(FPSCount);                    //  FPS Sayacını yükselt
```

```
LastTime :=ElapsedTime;
ElapsedTime :=GetTickCount() - DemoStart;     // gecen süre hesapla
ElapsedTime :=(LastTime + ElapsedTime) DIV 2;

glDraw();                    // Senaryoyu çiz
SwapBuffers(h_DC);           // Oynat
if GetTickCount-gecensure>=20000 then
…… // Buradan sonra sonuçları kaydedip uygulama kapatılmaktadır.
```

Yukarıdaki kod satırı ile, 3D animasyonun FPS değeri alınmakta ve 2 saniye sonra uygulama sonlandırılmaktadır. Uygulamanın ekran görüntüsü Şekil 3.30' daki gibidir.



Şekil 3.30. OPEN GL FPS Testi için uygulamanın ürettiği ekran görüntüsü.

3.2.6.7. Boyutlandırma Testi

Hafızaya alınan imaj dosyasının, matematiksel bir algoritma ile imaj çözünürlüğünün değiştirilmesi işlemidir. Aşağıda, uygulamada bu işlemi yapan kod satırı verilmiştir. Bu algoritma ile, yeni çözünürlüğe uygun şekilde, imaj dosyasındaki her pikselin yakınındaki piksellerin yeni değeri hesaplanmaktadır.

```
DstLine := Dst.ScanLine[0];
DstGap  := Integer(Dst.ScanLine[1]) - Integer(DstLine);
xP2 := MulDiv(pred(Src.Width), $10000, Dst.Width);
yP2 := MulDiv(pred(Src.Height), $10000, Dst.Height);
yP  := 0;
for y := 0 to pred(Dst.Height) do
begin
xP := 0;
SrcLine1 := Src.ScanLine[yP shr 16];
if (yP shr 16 < pred(Src.Height)) then
SrcLine2 := Src.ScanLine[succ(yP shr 16)]
else
SrcLine2 := Src.ScanLine[yP shr 16];
z2  := succ(yP and $FFFF);
iz2 := succ((not yp) and $FFFF);
for x := 0 to pred(Dst.Width) do
begin
t3 := xP shr 16;
z  := xP and $FFFF;
w2 := MulDiv(z, iz2, $10000);
w1 := iz2 - w2;
w4 := MulDiv(z, z2, $10000);
w3 := z2 - w4;
DstLine[x].rgbtRed := (SrcLine1[t3].rgbtRed * w1 +
SrcLine1[t3 + 1].rgbtRed * w2 +
```

```
SrcLine2[t3].rgbtRed * w3 + SrcLine2[t3 + 1].rgbtRed * w4) shr 16;
DstLine[x].rgbtGreen :=
 (SrcLine1[t3].rgbtGreen * w1 + SrcLine1[t3 + 1].rgbtGreen * w2 +
SrcLine2[t3].rgbtGreen * w3 + SrcLine2[t3 + 1].rgbtGreen * w4) shr 16;
DstLine[x].rgbtBlue := (SrcLine1[t3].rgbtBlue * w1 +
SrcLine1[t3 + 1].rgbtBlue * w2 +
SrcLine2[t3].rgbtBlue * w3 +
SrcLine2[t3 + 1].rgbtBlue * w4) shr 16;
Inc(xP, xP2);
end;
Inc(yP, yP2);
DstLine := pRGBArray(Integer(DstLine) + DstGap);
end;
```

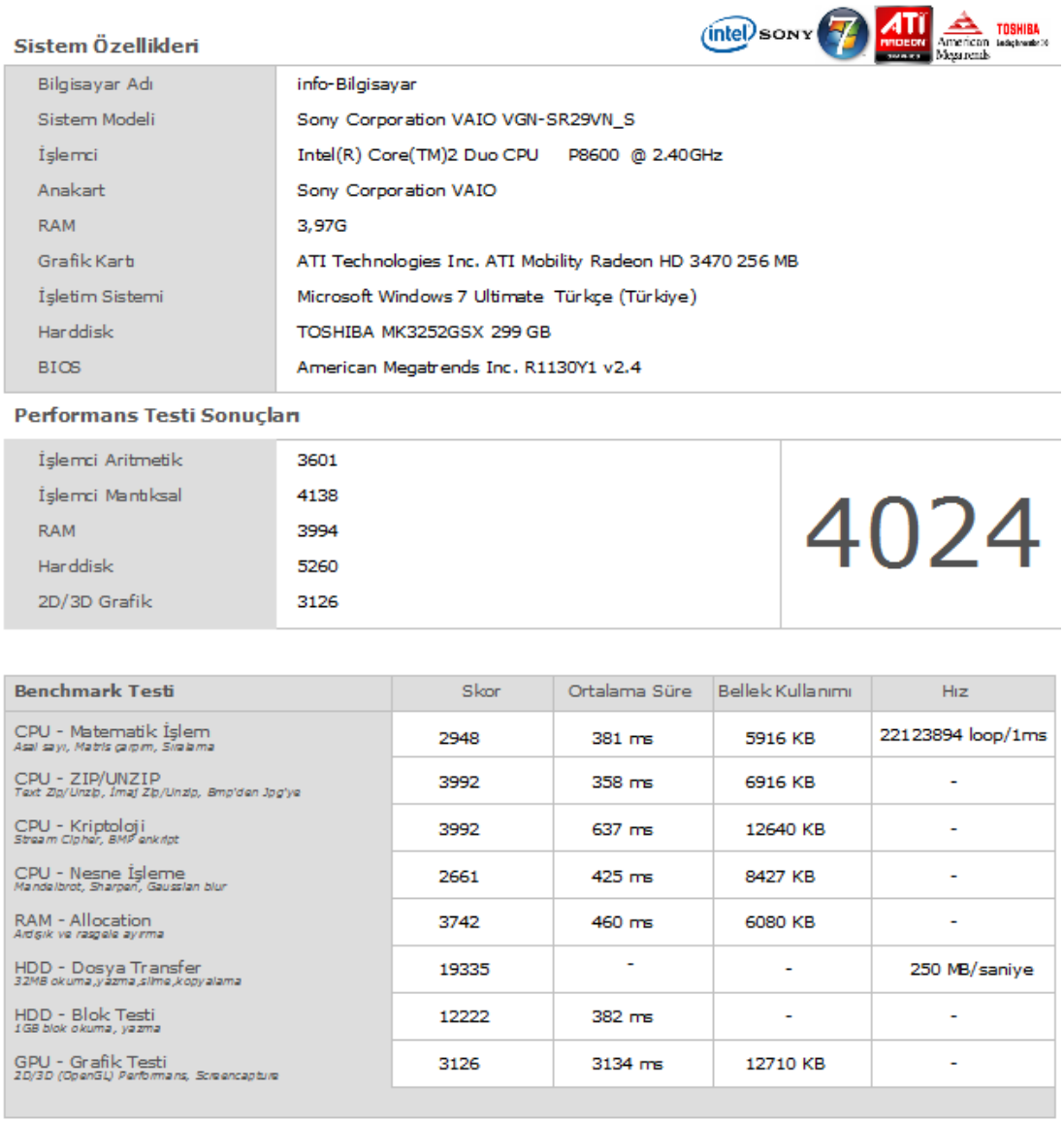
4. ARAŞTIRMA BULGULARI

4.1. Programın Ürettiği Test Sonucu

Geliştirilen uygulama test sonucunda, her ekranda ürettiği sonuçların ortalaması ile ;

- İşlemci aritmetik
- İşlemci mantıksal
- RAM
- Harddisk
- Grafik Kartı

skorları üretmekte, bu beş skorun ortalamasını alarak, test sonucunu tek bir skorla ifade etmektedir. Test sonucunda ayrıntılı skor tabelası, yazıcı çıktısı alınabilecek şekilde kullanıcıya QuickReport penceresinde sunulur (Şekil 4.1.).



Sistem Özellikleri

Bilgisayar Adı	info-Bilgisayar
Sistem Modeli	Sony Corporation VAIO VGN-SR29VN_S
İşlemci	Intel(R) Core(TM)2 Duo CPU P8600 @ 2.40GHz
Anakart	Sony Corporation VAIO
RAM	3,97G
Grafik Kartı	ATI Technologies Inc. ATI Mobility Radeon HD 3470 256 MB
İşletim Sistemi	Microsoft Windows 7 Ultimate Türkçe (Türkiye)
Harddisk	TOSHIBA MK3252GSX 299 GB
BIOS	American Megatrends Inc. R1130Y1 v2.4

Performans Testi Sonuçları

İşlemci Aritmetik	3601
İşlemci Mantıksal	4138
RAM	3994
Harddisk	5260
2D/3D Grafik	3126

4024

Benchmark Testi	Skor	Ortalama Süre	Bellek Kullanımı	Hız
CPU - Matematik İşlem *Asal sayı, Matris çarpım, Sıralama*	2948	381 ms	5916 KB	22123894 loop/1ms
CPU - ZIP/UNZIP *Text Zip/Unzip, İmaj Zip/Unzip, Bmp'den Jpg'ye*	3992	358 ms	6916 KB	-
CPU - Kriptoloji *Stream Cipher, BMP şifreleme*	3992	637 ms	12640 KB	-
CPU - Nesne İşleme *Mandelbrot, Sharpen, Gaussian blur*	2661	425 ms	8427 KB	-
RAM - Allocation *Ardışık ve rasgele ayırma*	3742	460 ms	6080 KB	-
HDD - Dosya Transfer *32MB okuma,yazma,silme,kopyalama*	19335	-	-	250 MB/saniye
HDD - Blok Testi *1GB blok okuma, yazma*	12222	382 ms	-	-
GPU - Grafik Testi *2D/3D (OpenGL) Performans, Screencapture*	3126	3134 ms	12710 KB	-

Şekil 4.1. SONY VAIO SR29VN Notebook Bilgisayarda tüm testlerin sonuçlarının QuickReport ekranında gösterimi. (Intel P8600 2Core CPU, 4GB RAM, ATI Radeon HD3470 GPU 256MB VRAM)

Bu pencere kapatıldıktan sonra kıyaslama ekranı kullanıcıya gösterilmektedir. Kıyaslama ekranında, kullanıcı kendi test sonucunu, veri tabanındaki önceden yapılmış test sonuçları ile karşılaştırabilmekte ve grafiksel ifadelerini inceleyebilmektedir.

4.1.1. Sistem Özellikleri

Ana ekranda gösterilen bilgisayar adı, sistem modeli, anakart, işlemci, ram, grafik kartı, işletim sistemi, harddisk ve bios marka ve modelleri bu bölümde gösterilmektedir (Şekil 4.1.).

4.1.2. Performans Testi Sonuçları

Sol tarafta işlemci aritmetik ve mantıksal, bellek, harddisk ve grafik kartı performans skorları gösterilmekte, sağ tarafta da bu beş skorun ortalaması bilgisayarın toplam skoru olarak gösterilmektedir (Şekil 4.1.).

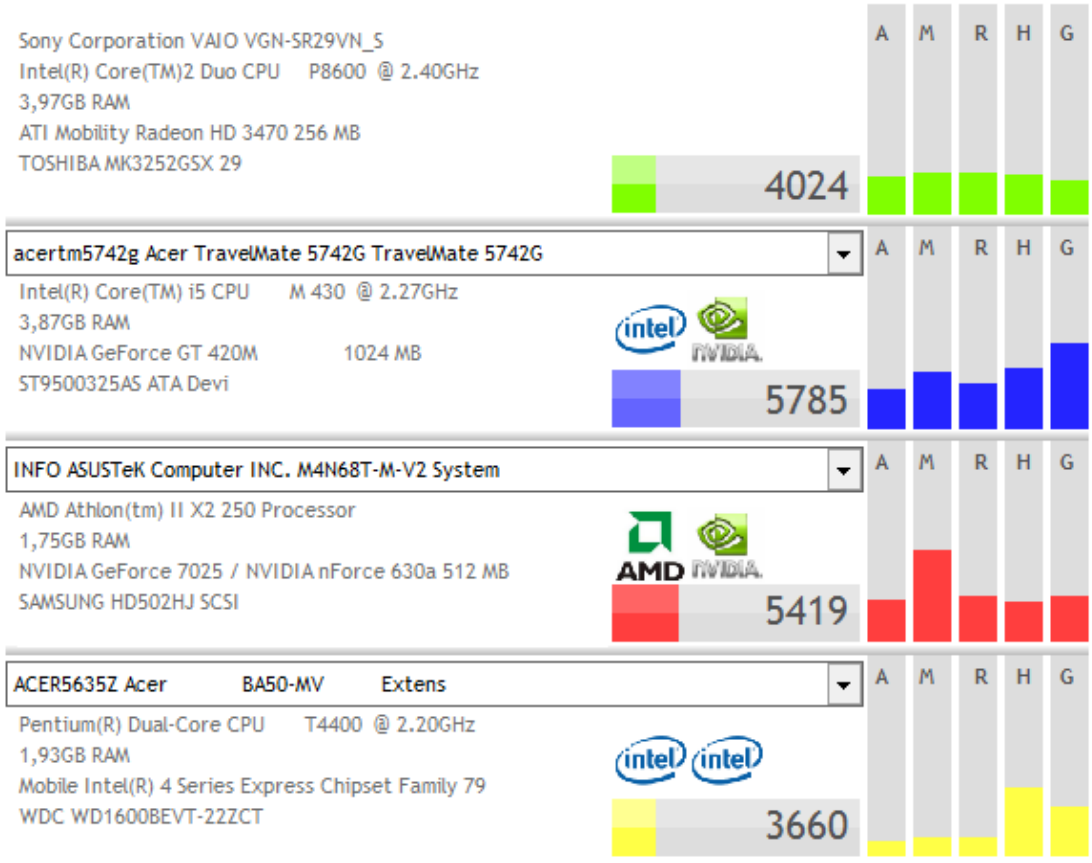
4.1.3. Benchmark Testi

Benchmark testi sonrasında, programın ürettiği skorlar yazdırılabilir ekranda detaylı olarak gösterilmektedir (Çizelge 4.1.). Ana ekranda test sonucunda, mevcut sistemle farklı sistemleri karşılaştırmak için kıyaslama ekranı gösterilmektedir (Şekil 4.2.).

Çizelge 4.1. Programın ürettiği skorların gösterimi.

Skor	Ortalama Süre	Bellek Kullanımı	Hız
3180	368 ms	5988 KB	22935780 loop/1ms
3780	347 ms	7038 KB	-
4179	626 ms	12688 KB	-
2786	418 ms	8459 KB	-
3949	435 ms	6241 KB	-
18350	-	-	234 MB/saniye
12812	359 ms	-	-
3817	3832 ms	12912 KB	-

- **Skor:** Belirtilen algoritmalar sonucunda verilen sayısal puan.
- **Ortalama Süre:** Belirtilen algoritmaların ortalama bitirilme zamanı.
- **Bellek Kullanımı:** Belirtilen algoritmaların çalışma zamanında kullandığı kilobayt cinsinden ortalama bellek miktarı.
- **Hız:** İlk satırda gösterilen loop / 1ms değeri; işlemcinin 1 milisaniyede kaç kez döngü çevirebildiğini gösterir. Bu değer genellikle işlemcinin frekans hızına yakın değerde çıkmalıdır. Altıncı satırda gösterilen MB / saniye ile ifade edilen değer, harddisk testlerinde ortalama ölçülen veri aktarım hızını göstermektedir. Veri aktarım hızı, harddiskin üretici değeriyle aynı değildir. Bu değer harddisk test algoritmasının ölçtüğü ortalama değerdir.



Şekil 4.2. Uygulamanın, test sonucunda sunduğu kıyaslama ekranı.

4.2. Test Sonuçlarının Diğer Benchmark Yazılımları ile Karşılaştırılması

Geliştirilen uygulamanın ürettiği skorların tutarlılığını ölçmek için uygulama birkaç farklı bilgisayarda, farklı benchmark yazılımı ile birlikte çalıştırılmalı ve tüm bu uygulamaların farklı kategorilerde verdiği skorlar karşılaştırılmalıdır.

4.2.1. Çoklu Bileşene Dayalı Karşılaştırma

Benchmark yazılımları, test algoritmalarının ürettiği skorları, tüm sistemi puanlayarak veya tek bir bileşeni puanlayarak kullanıcıya sunabilir. Geliştirilen uygulamayı diğer programlarla kıyaslamak için, hem tek bileşene dayalı, hem de tüm sistem ortalama skorları kıyaslamak gerekir.

GeekBench yazılımı, tüm sisteme ortalama skor verdiği için, tüm sistem puanlarını geliştirilen uygulama ile kıyaslamak için uygundur. PassMark CPU ve GPU test sonuçları da, geliştirilen uygulamanın CPU ve GPU sonuçlarını karşılaştırmak için uygundur.

4.2.1.1. Primate Labs GeekBench ile Karşılaştırma

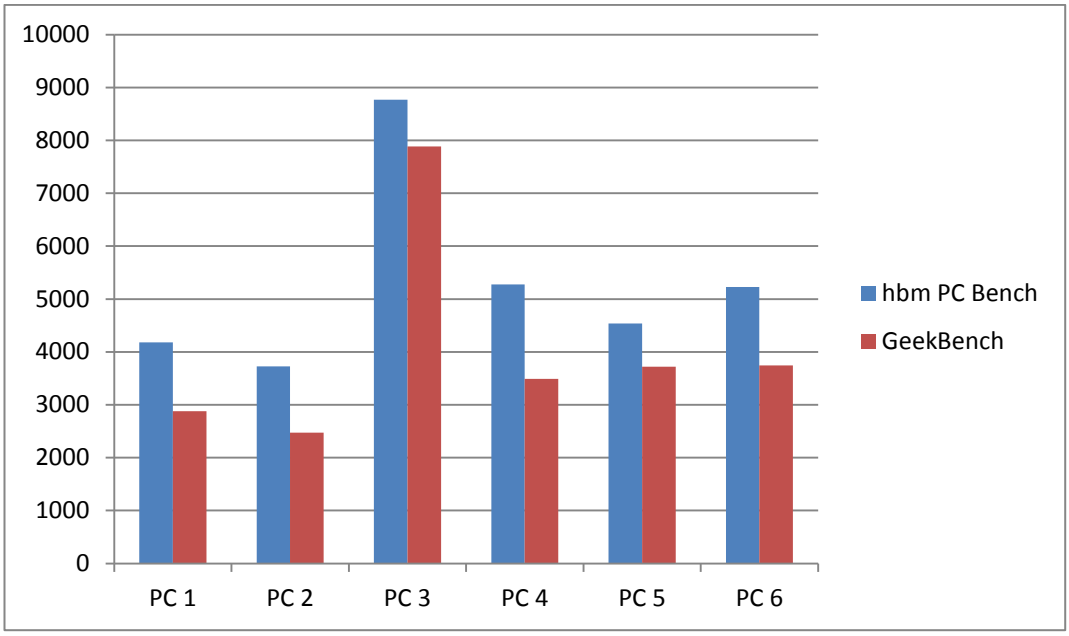
Primate Labs firmasının ürettiği, Geekbench yazılımının 2.2.7 versiyonu kullanılarak,6 farklı bilgisayar üzerinde skor karşılaştırması yapılmıştır. Kullanılan bilgisayarlar ve sonuçlar Çizelge 4.2. ve şekil 4.3.' teki gibidir;

1. **PC 1:**SONY VAIO SR29VN Notebook (Intel P8600 CPU, 4 GB 800Mhz RAM, 256MB ATI HD3470 GPU)
2. **PC 2:** HP Compaq CQ61 Notebook (Intel Dual Core T4300 CPU, 2 GB 667Mhz RAM, 512MB Nvidia G103M GPU)
3. **PC 3:** Masaüstü PC (Intel Core i5 2400, 8 GB 1333 Mhz RAM, 1 GB GeForce GTX460 GPU)
4. **PC 4:** Exper Notebook (Intel Core i3 M370, 2GB DDR3 1333Mhz, Intel HD Graphics GPU)

5. **PC 5:** Packard Bell NEW91 Notebook (Intel Core i3 M350 CPU, 3GB DDR3 1333Mhz RAM, 1GB Intel GT320M GPU)
6. **PC 6:** Masaüstü PC (AMD X2 250 CPU, 2 GB 1333Mhz RAM, Nvidia Entegre GPU)

Çizelge 4.2. Uygulamanın ve Primate Labs Geekbench yazılımının 6 farklı platformda verdiği sonuçlar. (Sayısal değer arttıkça performans değeri artmaktadır.)

	PC 1	PC 2	PC 3	PC 4	PC 5	PC 6
Geekbench integer	3388	2903	7723	3850	3649	3847
PC Bench Aritmetik	3763	2980	6722	4046	3991	3819
Geekbench floating point	2943	2372	8674	3580	4298	4637
PC Bench Mantıksal	4108	3595	8447	5085	5159	9015
Geekbench RAM	2186	2063	6856	2936	3108	2685
PC Bench RAM	4204	3712	6887	4248	4181	4634
Geekbench Toplam	2881	2473	7886	3489	3718	3743
PC Bench Toplam	4179	3729	8769	5273	4538	5228



Şekil 4.3. Geliştirilen uygulamanın sonuçlarının PrimateLabs GeekBench ile kıyaslanması

İki uygulama arasında kullanılan algoritmalar ve puanlama mekanizması farklılık gösterse bile, farklı platformlarda yapılan testlerin sonucu 2 boyutlu düzlemde parabolik grafikle ifade edildiği zaman, iki eğrinin birbirlerine eğimleri göz ardı edilecek kadar küçüktür. Bu durumda geliştirilen uygulamanın, uluslararası piyasada kabul görmüş ticari uygulamalara yakın sonuçlar ürettiği görülmektedir.

4.2.2. Tek Bileşene Dayalı Karşılaştırma

Geliştirilen uygulama, her bileşen için kullanılan benchmark algoritmalarının sonuçlarına göre ortalama skor üretebilmektedir. Ancak benchmark yazılımları, özellikle dizüstü bilgisayarlarda, tek bileşene dayalı performans ölçümü yaparken bazı faktörlerden etkilenirler;

- Bilgisayarın enerjisini sağlayan güç kaynağı (batarya, elektrik kaynağı)
- İşletim sisteminin güç yönetim yazılımından kaynaklanana performans kısıtlamaları
- Bilgisayarın diğer bileşenlerinden kaynaklanan performans değişiklikleri

- Arka planda çalışan uygulamaların işlemci ve bellek yazmaçlarında yer tutması

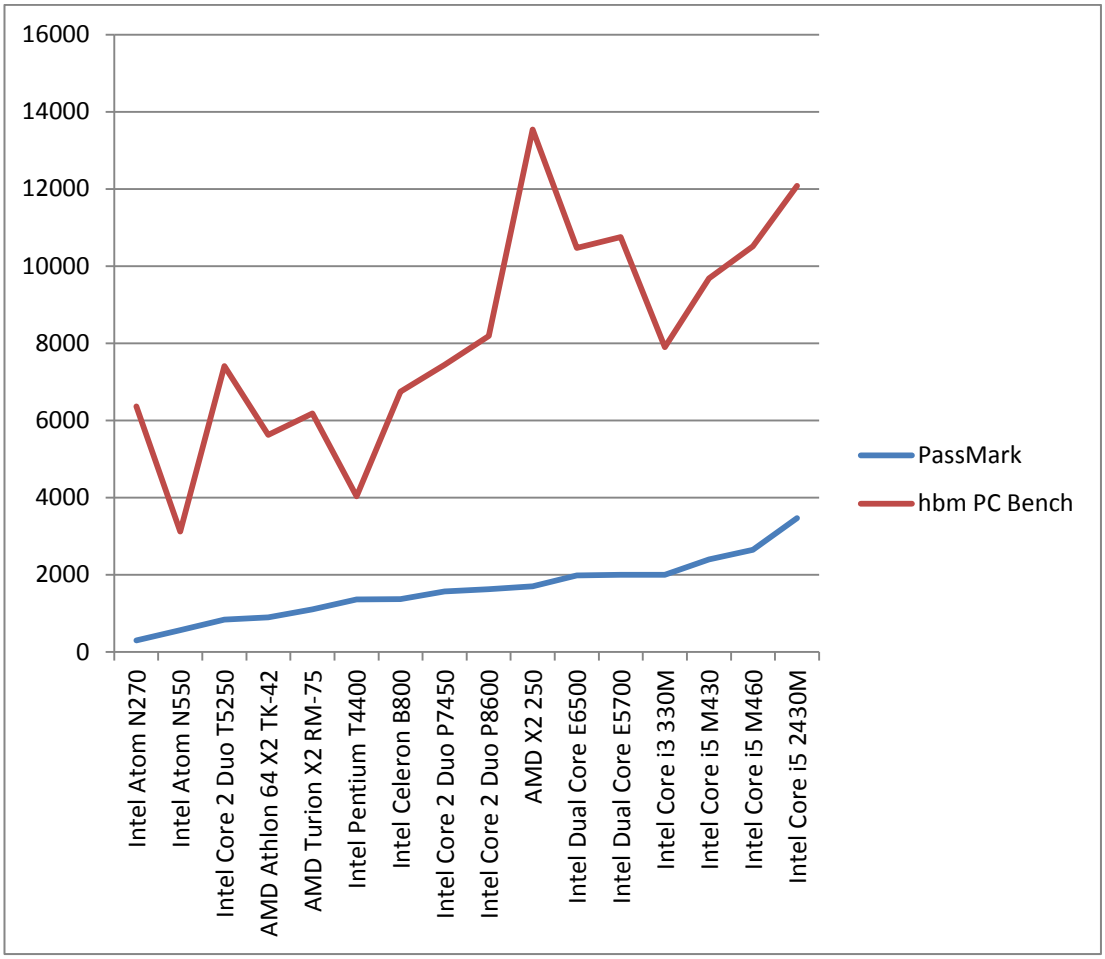
4.2.2.1. PassMark CPU Benchmark Sonuçları ile Karşılaştırma

PassMark yazılım, merkezi Avusturalya, Sydney' de bulunan bir yazılım geliştirme firmasıdır. 1998 yılında kurulan firma, geliştirdiği benchmark araçları ile, belirli bir standart içinde bugüne kadar, 300 ün üzerinde farklı işlemcinin kullanıldığı 20,000' den fazla sistemin benchmark işlemini yapmış ve benchmark sonuçlarını web sitesinde yayınlamıştır.

Bu çalışmada geliştirilen uygulamanın ürettiği CPU mantıksal ve CPU aritmetik sonuçlarının toplamı, PassMark web sitesinde yayınlanan sonuçları ile çizgisel grafik yardımı ile kıyaslanmıştır (Çizelge 4.3, Şekil 4.4.).

Çizelge 4.3. PassMark ve geliştirilen programın ürettiği CPU testi sonuçları. (Sayısal değer arttıkça CPU performans değeri artmaktadır.)

İşlemci Modeli	PassMark Skoru	hbm PC Bench skoru mantıksal + aritmetik
Intel Atom N270	304	6365
Intel Atom N550	569	3120
Intel Core 2 Duo T5250	836	7409
AMD Athlon 64 X2 TK-42	900	5624
AMD Turion X2 RM-75	1106	6184
Intel Pentium T4400	1366	4036
Intel Celeron B800	1369	6745
Intel Core 2 Duo P7450	1567	7439
Intel Core 2 Duo P8600	1628	8191
AMD X2 250	1701	13543
Intel Dual Core E6500	1987	10470
Intel Dual Core E5700	1997	10756
Intel Core i3 330M	1999	7900
Intel Core i5 M430	2402	9683
Intel Core i5 M460	2651	10511
Intel Core i5 2430M	3469	12079



Şekil 4.4. Geliştirilen uygulamanın sonuçlarının PassMark istatistikleri ile kıyaslanması.

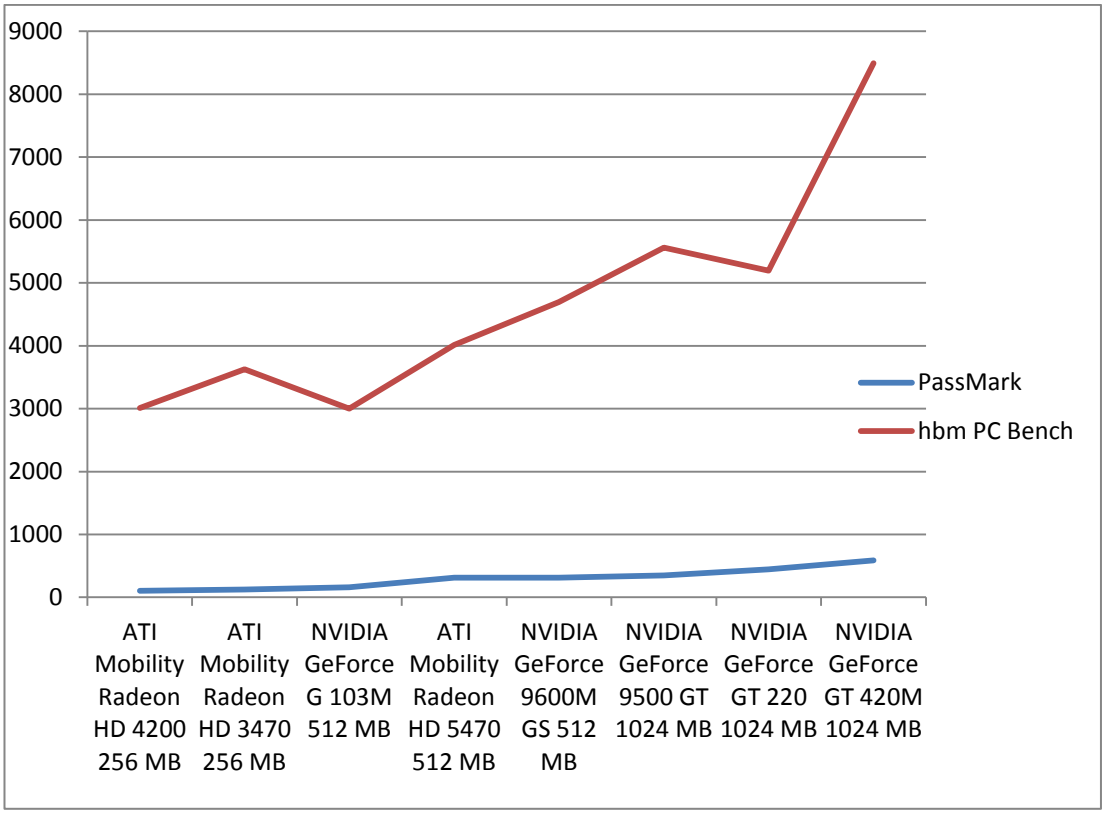
Grafikteki işlemci modelleri, soldan sağa doğru, PassMark benchmark sonuçlarına göre, küçükten büyüğe doğru dizilmiştir. (Şekil 4.4.) Kırmızı çizgi, geliştirilen uygulamanın ürettiği benchmark sonuçlarını göstermektedir. Kırmızı çizgi ve mavi çizgi arasındaki farklılığın olası sebepleri, önceki başlıkta anlatılmaktadır.

4.2.2.2. PassMark GPU Benchmark Sonuçları ile Karşılaştırma

Bu çalışmada geliştirilen uygulamanın ürettiği GPU sonuçlarının toplamı, PassMark web sitesinde yayınlanan sonuçları ile çizgisel grafik yardımı ile kıyaslanmıştır (Çizelge 4.4., Şekil 4.5.).

Çizelge 4.4. PassMark ve geliştirilen programın ürettiği GPU testi sonuçları. (Sayısal değer arttıkça GPU performans değeri artmaktadır.)

İşlemci Modeli	PassMark Skoru	hbm PC Bench skoru mantıksal + aritmetik
ATI Mobility Radeon HD 4200 256 MB	105	3009
ATI Mobility Radeon HD 3470 256 MB	127	3627
NVIDIA GeForce G 103M 512 MB	161	3002
ATI Mobility Radeon HD 5470 512 MB	312	4012
NVIDIA GeForce 9600M GS 512 MB	314	4697
NVIDIA GeForce 9500 GT 1024 MB	349	5559
NVIDIA GeForce GT 220 1024 MB	444	5197
NVIDIA GeForce GT 420M 1024 MB	587	8494



Şekil 4.5. Geliştirilen uygulamanın sonuçlarının PassMark istatistikleri ile kıyaslanması.

5. SONUÇ

Benchmark yazılımlarının en büyük eksikliği, tek yönlü çalışmaları ve çok sayıda dış etkene bağlı olarak hatalı sonuçlar üretebilmeleridir. Birçok benchmark yazılımı, sadece tek bir bilgisayar bileşenini sınırlı sayıda test algoritması ile test etmektedir. Sonuçların güvenilirliğini ölçmek için kullanıcı birçok farklı uygulamayı çalıştırıp, sonuçları doğrulamak durumunda bırakılmaktadır. Bir bileşene benchmark uygulanırken, oluşturulan puanlama, sistemin diğer bileşenlerinin başarımından etkilenmektedir.

Benchmark yazılımlarının birçoğu internet üzerinde ücretsiz temin edilebilmektedir. Üreticilerin baz aldığı benchmark yazılımları ise genellikle yüksek ücretler karşılığında satılmaktadır. Dolayısıyla, tüketici, kendisi yapmadığı bir testi, üreticinin verdiği sonuçlara inanmak durumunda kalarak kabul etmekte ve bu sonuçlara göre ürün satın alma tercihi yapmaktadır. Bu durum haksız rekabete yol açmaktadır.

Geliştirilen uygulama; kabul görmüş yazılımlarla kıyaslandığında, tutarlı skorlar üretmiş, uygulamanın güvenilirliği ispatlanmıştır (Şekil 4.3. Şekil 4.4. Şekil 4.5.). Ayrıca, uygulama SiSoft Sandra benchmark raporlarına benzer şekilde, AMD marka işlemcilerin mantıksal test sonuçlarının, eşdeğer Intel marka işlemcilere göre yüksek çıktığını, Intel marka işlemcilerin aritmetik performansının da, eşdeğer AMD marka işlemcilere göre daha yüksek olduğunu da, ürettiği sonuçlarla ispatlamıştır (O'flaherty vd., 2003).

6. KAYNAKLAR

Baş, Mehmet., 2012. Çağdaş Yönetim Kuramlarından Biri Benchmarking,, Gazi Üniversitesi İ.İ.B.F.

Başlıgil, Hüseyin., 2009. Sistem Analizi.

Bird, Sarah., Phansalkar, Aashish., John, Lizy K., Mericas, Alex., Indukuru, Rajeev., 2012. Performance Characterization of SPEC CPU Benchmarks on Intel's Core Microarchitecture based processor, University of Texas at Austin.

Curnow, H. J., Wichmann, B. A., 1976, A Synthetic Benchmark, The Computer Journal.

Dongarra, J. J., 1983. Performance of various computers, Computer Architecture News.

Hogan, Thom., 2003. Sharpening 101.

Kapoor, Manish., 2012. Benchmarks Software – Computer Benchmarks.

Li, Jing., 2009. ABC Macro and Performance Chart with Benchmarks Annotation, , AQAF, Birmingham, AL.

Mah Ung, Gordon., Case, Loyd, 2010. How to Properly Benchmark Your PC.

Middela, Sreekanth Reddy., 2008. Benchmark Macro %COMPARE, MaxisIT Inc.

Moore, Ronald., 2008. Benchmarking 100 Success Secrets.

Nambiar, Raghunath., 2010. Transaction Processing Performance Council, State of the Council.

Okulicka, Felicja., 2010. Paralel Programming Lecture 14 Computer Performance.

O'flaherty, Doug., Goddard, Michael, 2003. AMD Opteron™ Processor Benchmarking for Clustered Systems.

Pabeda, Patrick., Wijaya, Marcel., Sari Riri Fitri., 2006. Designing Virtual Reality on Grid Computing Platform, Faculty of Engineering University of Indonesia

Paris, Sylvain., 2010. Naïve Image Smoothing: Gaussian Blur.

Patel, Pranit., Wong, Jeff., Tatikonda, Manisha., Marczewski, Jarek., 2009. JPEG Compression Algorithm Using CUDA, Department of Computer Engineering, University of Toronto.

Rahmani, Sheida., Strait, Melissa., Merkurjev, Daria., Moeller, Michael., Wittman, Todd., 2008. An Adaptive IHS Pan-sharpening Method.

Sheth, Khushboo., 2005. Performance And Power Benchmarking.

Smith, Matt., 2010. The 5 Best Free Benchmark Programs for Windows.

Weicker, Reinhold P., 1991. A detailed look at some popular benchmarks. Parallel Computing.

Weiderman, Nelson., 1989. Hartstone: Synthetic Benchmark, Requirements for Hard Real-Time Applications.

York, Richard., 2002. Benchmarking in context: Dhrystone, By, ARM Ltd.

Yüksel, Öznur., 2003. Yönetim Fonksiyonları, Girişimciler İçin İşletme Yönetimi.

Zhang, Xiaolan., 2001. Application-Specific Benchmarking A thesis presented by To The Division of Engineering and Applied Sciences, Harvard University Cambridge, Massachusetts.

İnternet Kaynakları

http://en.wikipedia.org/wiki/Gaussian_blur. Erişim Tarihi: 12.02.2012

http://en.wikipedia.org/wiki/Mandelbrot_set. Erişim Tarihi: 16.03.2012.

http://en.wikipedia.org/wiki/SPEC. Erişim Tarihi: 12.02.2012

http://en.wikipedia.org/wiki/Stream_cipher. Erişim Tarihi: 10.01.2012

http://tr.wikipedia.org/wiki/Delphi_%28programlama_dili%29.
Erişim Tarihi: 07.03.2012.

http://tr.wikipedia.org/wiki/JPEG. Erişim Tarihi: 18.03.2012

http://tr.wikipedia.org/wiki/Paint_Shop_Pro. Erişim Tarihi: 19.03.2012.

http://www.mersenne.org/reportbenchmarks. Erişim Tarihi: 20.02.2012

http://www.mindcontrol.org/~hplus/graphics/ogl-perf.html.
Erişim Tarihi: 18.02.2012.

ÖZGEÇMİŞ

Kişisel Bilgiler

Adı ve Soyadı : Hüseyin Bilal MACİT

Doğum Yeri: Burdur

Doğum Yılı : 1982

Medeni Hali: Evli

Eğitim Durumu:

Lise: 1996 – 1999 / Burdur Anadolu Lisesi

Lisans: 2000-2004 / Çanakkale Üniversitesi – Mühendislik Mimarlık Fakültesi - Bilgisayar Mühendisliği Bölümü

Yüksek Lisans: 2008-?/ Süleyman Demirel Üniversitesi – Fen Bilimleri Enstitüsü - Bilgisayar Mühendisliği Bölümü

Yabancı Dil(ler) ve Düzeyi:

İngilizce / İyi

Almanca / Başlangıç