T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

TEST VEKTÖRÜ KULLANAN GERÇEK ZAMANLI KIYASLAMA UYGULAMA TAKIMI

YÜKSEK LİSANS TEZİ

Metin KUZHAN

Enstitü Anabilim Dalı	**:**	**BİLGİSAYAR VE BİLİŞİM MÜHENDİSLİĞİ**
Tez Danışmanı	**:**	**Dr. Öğretim Üyesi Veysel Harun ŞAHİN**

Haziran 2019

T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

TEST VEKTÖRÜ KULLANAN GERÇEK ZAMANLI KIYASLAMA UYGULAMA TAKIMI

YÜKSEK LİSANS TEZİ

Metin KUZHAN

Enstitü Anabilim Dalı : BİLGİSAYAR VE BİLİŞİM MÜHENDİSLİĞİ

Bu tez 10.06.2019 tarihinde aşağıdaki jüri tarafından oybirliği / oyçokluğu ile kabul edilmiştir.

Dr. Öğr. Üyesi Veysel Harun ŞAHİN Jüri Başkanı	Prof.Dr. Celal ÇEKEN Üye	Dr. Öğr. Üyesi Bahadır HİÇDURMAZ Üye

BEYAN

Tez içindeki tüm verilerin akademik kurallar çerçevesinde tarafımdan elde edildiğini, görsel ve yazılı tüm bilgi ve sonuçların akademik ve etik kurallara uygun şekilde sunulduğunu, kullanılan verilerde herhangi bir tahrifat yapılmadığını, başkalarının eserlerinden yararlanılması durumunda bilimsel normlara uygun olarak atıfta bulunulduğunu, tezde yer alan verilerin bu üniversite veya başka bir üniversitede herhangi bir tez çalışmasında kullanılmadığını beyan ederim.

Metin KUZHAN

30.04.2019

TEŞEKKÜR

Yüksek lisans eğitimim boyunca değerli bilgi ve deneyimlerinden yararlandığım, her konuda bilgi ve desteğini almaktan çekinmediğim, araştırmanın planlanmasından yazılmasına kadar tüm aşamalarında yardımlarını esirgemeyen, teşvik eden, aynı titizlikte beni yönlendiren değerli danışman hocam Dr. Öğretim Üyesi Veysel Harun ŞAHİN'e teşekkürlerimi sunarım.

Yapılan tez çalışması Sakarya üniversitesi, Bilgisayar ve Bilişim Fakültesi, Yazılım Mühedisliği Bölümü'ndeki Gerçek Zamanlı Sistemler Araştırma Laboratuvarı tarafından desteklenmektedir.

İÇİNDEKİLER

SİMGELER VE KISALTMALAR LİSTESİ

GPOS	: Genel amaçlı işletim sistemi
IoT	: Nesnelerin Interneti
RTEMS	: Çok işlemcili sistemler için gerçek zamanlı çalıştırıcı
RTOS	: Gerçek zamanlı işletim sistemi
WCET	: En kötü yürütme süresi

ŞEKİLLER LİSTESİ

TABLOLAR LİSTESİ

ÖZET

Anahtar kelimeler: RTEMS, Gerçek zamanlı sistem, Kıyaslama Uygulama Takımı

Gelişen teknoloji ile beraber günlük hayatımızdaki ve endüstrideki donanım cihazlarının kontrol edilmesini ihtiyacı ortaya çıkmıştır. Bu problemin giderilmesi amacıyla işletim sistemleri adı verilen yazılımlar piyasaya sürülmüştür.

Elektronik cihazlardaki hızlı gelişim ve küçülme daha küçük boyutlu cihazların üretilmesini ve kullanmasını sağlamıştır. Bundan dolayı ortaya çıkan küçük boyutlu cihazları yönetmek adına işletim sistemleri yetersiz kalmıştır. Küçük ölçekli cihazlarda veya zamanın kritik olduğu sistemlerde dış ortama cevap verebilen sistemleri yönetecek işletim sistemleri yazılmıştır. Bu problemi üstesinden gelmek için gömülü sistemlerde çalışmak üzere gerçek zamanlı işletim sistemleri tasarlanmıştır. Böylece sistemin kullanıcıya ve dış ortama cevap üretmek için oluşacak olan kritik zamanlı görevler yerine getirilebilmiştir.

Bu çalışmada gerçek zamanlı bir işletim sistemi olan RTEMS ile genel amaçlı bir işletim sistemi olan Xubuntu arasında ölçüm tabanlı en kötü yürütme süresini bulmaya yardımcı olacak bir kıyaslama takımı oluşturuldu. Test Vektörü alan kıyaslama uygulama takımı (MBBench)'nin en önemli özelliği farklı girdi değerlerine göre aynı programların farklı platformlar için çalışma sonuçları karşılaştırabilecektir.

MBBench bilgisayar bilimindeki iyi bilinen algoritmalardan seçilerek oluşturulmuş bir kıyaslama takımıdır. MBBench hem RTEMS hem de Xubuntu üzerinde çalıştırabilecek şekilde tasarlanmıştır. Kıyaslama takımının her iki platform için yazılan uygulamaları C dili ile kodlamıştır. Yapılan çalışma ile farklı girdi değerlerine göre farklı sonuçlar üretebilen bir kıyaslama takımı olan MBBench elde edilmiştir.

REAL TIME BENCHMARK SUITE USING TEST VECTOR

SUMMARY

Keywords: RTEMS, Real time system, Benchmark Suite

With the developing technology, the need to control the hardware devices in our daily life and industry has emerged. In order to solve this problem, the so-called operating systems were launched.

The rapid development and downsizing of electronic devices has led to the production and use of smaller devices. Therefore, operating systems have been insufficient to manage the resulting small devices. Operating systems have been written to manage systems that can respond to the external environment in small-scale devices or in systems where time is critical. To overcome this problem, real-time operating systems are designed to work on embedded systems. Thus, critical time tasks that will occur in order to produce response to the user and the external environment of the system have been fulfilled.

In this study, a benchmark suite was created to help find to worst measurement-based execution time between the real-time operating system RTEMS and the general-purpose operating system Xubuntu. The most important feature of real time benchmark suite using test vector (MBBench) is that it will be able to compare the results of the same programs to different platforms based on different input values.

MBBench is a benchmark suite that has been selected from well-known algorithms in computer science. MBBench is designed to run on both RTEMS and Xubuntu. The applications of the benchmarking team for both platforms were coded in C language. In this study, MBBench, a benchmark which can produce different results according to different input values, was obtained.

BÖLÜM 1. GİRİŞ

Gerçek zamanlı uygulamalar havacılık, nükleer güç santralleri ve hava yolu reservasyonları gibi çeşitli alanlarda kullanılmaktadır. Bu tip uygulamaların kullanıldıkları alan itibariyle hayati önem taşıyan sonuçlar üretmeleri gerekmektedir. Uygulamaların çıktısı olan verilerin zamanında üretilmelidir ve sonuçlar güvenilir olmalıdır [1].

Gerçek zamanlı sistemlerde uygulamaların çalışma zamanı tespit edilmelidir. Program gerçek zamanlı bir sistemde kullanılmadan önce en kötü yürütme süresinin ne kadar olacağın bilinmesi zorunlu hale gelmiştir. En kötü yürütme süresi bulmak için çalışmalar yapılmaktadır. Yüksek seviyeli programlama dilleri ile yazılan kıyaslama uygulamaları ile en kötü yürütme süresi (WCET) tahmin edilebilmektedir [2].

RTEMS, gerçek zamanlı uygulamaların çalıştırılabileceği gerçek zamanlı bir işletim sistemidir. İlk kullanılmaya başlandığında yazılan gerçek zamanlı uygulamalar Ada programlama dilini kullanarak yazılmıştır ve yazılmış olan uygulamalar tek işlemcili sistemler için kullanılmıştır. Teknolojik gelişmeler ile birlikte çok çekirdekli ve çok işlemcili sistemler kullanılmaya başlanmıştır. Genel olarak bu sistemlerde kullanılan kıyaslama uygulamaları C ve Java programlama dili kullanılarak yazılmıştır [3].

Bu çalışmada test vektörü kullanan gerçek zamanlı bir kıyaslama uygulama takımı yazılması hedeflenmiştir. Bundan dolayı ilk olarak kıyaslama uygulamaları C programlama dili ile yazılmış ve XUBUNTU 17.10 işletim sistemi üzerinde derlenip

çalıştırılmıştır. Linux tarafında oluşturulan kıyaslama uygumaların bitmesinden sonra RTEMS, Xubuntu işletim sistemi üzerinde kuruldu ve gerekli test işlemleri gerçekleştirildi. RTEMS gerçek zamanlı işletim sisteminde çalıştırılabilecek şekilde kaynak kodlar çok değiştirilmeden uygulamaların test işlemleri sparc simülatörü üzerinde gerçekleştirildi. RTEMS ve Xubuntu üzerinde test edilecek uygulamalar dışarıdan girdi alabilecek algoritmalardan seçilmiştir. Çalışma sonucunda en kötü yürütme süresi bazlı ölçüm yapabilen ve farklı test vektörü çeşitlerini alabilen bir gerçek zamanlı kıyaslama takımı elde edilmiştir.

BÖLÜM 2. GERÇEK ZAMANLI SİSTEMLER

Gerçek zamanlı sistemler çevresindeki oluşan olaylara karşı belli bir zaman içerisinde tepki vermek zorunda olan bilgisayar sistemleridir. Bu tip sistemlerin çalışmasının doğruluğu üretilen sonuçların yanında sonuçların verildiği zamanında büyük bir önemi vardır. Gerçek zamanlı sistemin geç cevap vermesi durumda üretilen sonuç kullanılamaz hale gelebilir veya tehlikeli sonuçlar oluşabilir.

Genel anlamda sistemleri kontrol eden gerçek zamanlı bilgisayarlar sistemlere gömülerek kullanılır. Örnek olarak gömülü sistemlerin bulunduğu yerlere şunlar verilebilir: [4]

- Kamera
- Navigasyon
- Endüstriyel Robot
- Araba
- Uçak

Gerçek zamanlı bir kontrol sisteminin dışarıdan veri sağlamak için sensörler ve kameraları kullanmaktadır. Sistemin dışarıya çıktı verebilmek için aktuatörler ve kameraları kullanmaktadır [1].

Sensörler ve Aktüatörler dışarıdan aldıkları fiziksel sinyalleri dijital forma dönüştürür. Ayrıca dijital sinyalleri de fiziksel forma dönüştürebilir. Bu sayede gerçek zamanlı sistem dışarıdan veri alabilir veya dışarıya çıktı verebilir. Böylece gerçek zamanlı bir sistem çevresiyle etkileşime geçebilir.

Gerçek zamanlı bir bilgisayar sistemi çevresindeki uyarıcıların yaptığı uyarmalara göre sonuç üretmek zorunda olan bir sistemdir. Üretilen sonuç çevrenin belirlediği bir zaman aralığı içinde olmalıdır. Sistemin ürettiği sonucun üretilmek zorunda olduğu anlık zamana son teslim zamanı(deadline) olarak ifade edilmektedir.

Gerçek zamanlı bir bilgisayar sistemi için iki tip son teslim zamanı karşımıza çıkmaktadır. Sistemin ürettiği cevap son teslim zamanından sonra kullanabiliyorsa bu tür son teslim zamanlarına yumuşak(soft) olarak ifade edilir. Aksi yönden düşünülürse sistemin ürettiği cevap son teslim zamanından sonra üretildiğinde telafisi mümkün olmayan sonuçlara sebep oluyorsa bu son teslim zaman tipinede katı(hard) olarak ifade edilmektedir.

Bir gerçek zamanlı bilgisayar sisteminde en az bir tane katı son teslim zamanına sahip olan sistemlere katı gerçek zamanlı bilgisayar sistemi olarak isimlendirilmektedir. Aksi yönden bakılacak olursa gerçek zamanlı bir bilgisayar sistemi herhangi bir katı son teslim zamanına sahip değil ise de yumuşak gerçek zamanlı sistem olarak isimlendirilmektedir [6].

Gerçek zamanlı sistemler görevleri yerine getirirken belirlenmiş olan kısıtlamalar vardır. Bu kısıtlamalardan bir tanesi de görevi yerine getirmenin yanında işin yapılma zamanı açısından değerlendirilmektedir. Buna göre gerçek zamanlı sistemlerde iki adet zaman kısıtlaması bulunmaktadır. Bunlar katı ve yumuşak zaman kısıtlamalarıdır. Gerçek bir sistem her iki kısıtlama türüde bulunabilir [7].

2.1. Gerçek zamanlı sistem tipleri

Gerçek zamanlı sistemler iki şekilde sınıflandıralabilir: [5]

- Yumuşak gerçek zamanlı sistemler
- Katı gerçek zamanlı sistemler

2.1.1. Yumuşak gerçek zamanlı sistemler

Yumuşak gerçek zamanlı sistemler, sisteminin verdiği bir hizmetin tamamlanmasının net olarak ne zaman olduğu belli olmayan sistemlerdir. Bu tür sistemlerde son teslim zamanından sonra sonuçlar üretilebilir. Üretilen sonucunun son teslim zamanından sonra olması kullanışsız olduğu manasına gelmez. Üretilen sonuçların kabul edilebilir olduğu bir aralık vardır. Sistemin kalitesi görevin tamamlanmasının son teslim zamanına ne kadar yakınsa o oranda daha hizmet kalitesi yüksek bir sistem olarak görülür.

Yumuşak gerçek zamanlı sistemlere aşağıdaki örnekler verilebilir: [6] [7]

- Dijital kamera
- GPS
- Kablosuz yönlendirici
- Cep telefonları
- Çevrim içi veritabanları

2.1.2. Katı gerçek zamanlı sistemler

Katı gerçek zamanlı sistem, sistemin vermesi gereken hizmetin belirlenen son teslim zamanını geçmesi durumunda telafisi mümkün olmayan etkilere sebep olan katı zaman kısıtlaması ya da kısıtlamaları bulunan sistemlerdir. Yerine getirmesi gerektiği hizmetin son teslim zamanından sonra tamamlanması kabul edilemeyen bir durumdur. Sistemin istenildiği gibi çalışmaması durumunda önemli derecede duruma göre mali veya hayati sonuçlar doğurabilir. Özetlemek gerekirse katı gerçek zamanlı sistemlerde hem hizmetin doğru olarak verilmesi hem de zamanında icra edilmesi gerekmektedir.

Katı gerçek zamanlı sistemlere aşağıdaki örnekler verilebilir: [6] [7]

- Kalp pili

- Füze savunma sistemleri
- Arabalardaki hava yastığı kontrol sistemi
- Motor kontrol sistemleri

2.2. Gerçek zamanlı işletim sistemi

Gerçek zamanl işletim sistemi (RTOS) belli olan zaman kısıtlamaları içinde donanım cihazları kontrol etmek için tasarlanmış bir işletim sistemidir. Mevcut RTOS'ların birçoğu gömülü sistemlerde kullanılmaktadır [8].

RTOS normal bir işletim sisteminden en önemli farkı verilen hizmetin hem doğru hem de zamansal olarak uygun olarak üretilmek zorundadır. Sıradan bir işletim sistemi sadece verdiği hizmetin doğru olarak verilmesi yeterlidir. Hizmetinin tamamladığı zamanın kritik bir önemi yoktur. Normal bir işletim sistemi verilen hizmetinin yani görevin tamamlanması sonsuz bir zaman içinde bitirebilir.

RTOS göstereceği davranışa karar vermek için özelleştirilmiş çizelgeleme algoritmalarını kullanır. Bu sayede cevap verilmek istenen olayın nasıl daha çabuk yapılacağına ve ne kadar sürede gerçekleşeğine ortaya çıkarmaya çalışır [9].

RTOS'ların ücretsiz ve ücretli olarak kullanabilir. Ücretli olarak LynxOS, ThreadX, VRTX, WindowsCE kullanılabilir. Ücretsiz olarak FreeRTOS, RTLinux kullanılabileceklere örnek olarak verilebilir [10].

RTOS'lar aslında birer işletim sistemidir. Günümüzde günlük hayatta kullandığımız bilgisayarları yönetmek için kullanılan işletim sistemleri ise genel amaçlı işletim sistemleri (GPOS) olarak adlandırılmaktadır [7].

GPOS ile RTOS lar ikisi de kavram olarak birer işletim sistemi olması rağmen taşıdıkları özellik bakımından birbirlerinden farklıdır [9]. GPOS'lara şunlar örnek verilebilir: Windows, Linux ve MacOS dur. RTOS'lara örnek olarak şunlar verilebilir: RTEMS, VxWorks, eCos, Zephyr.

2.2.1. Genel amaçlı ile gerçek zamanlı işletim sistemi arasındaki farklar

Çoğunlukla gömülü sistemlerde kullandığımız bir işletim sistemi RTOS ile günlük hayatımızdaki bilgisayarları yönetmek için kullandığımız GPOSlar birbirinden özellik bakımından farklılık göstermektedir. İkisi arasındaki en önemli fark dış olaylara verilen tepki süresidir. GPOS herhangi bir görevin tamamlanmasını garanti etmek zorunda değildir. Son teslim zamanı üzerinden değerlendirdiğimizde yumuşak gerçek zamanlı bir cevap verme özelliği gösterir. RTOS ise dış olaylara karşı hızlı ve yüksek oranda cevap vermesi gerekir. Özellikle katı gerçek zamanlı sistemlerde belli görevlerin tamanlanması süre olarak da garanti edilmelidir [11].

RTOS ve GPOS arasındaki farklar şu şekilde özetlenebilir:

- RTOS verilen görevin zamanında tamanlamasına odaklanmaktadır. GPOS ise verilen görevin zamanında tamanlanmasında ziyade birden çok görevin başarılması üzerine tasarlanmıştır.
- GPOS yapılacak görevlerin sıralanması konusunda herhangi bir algoritma kullanabilmektedir. Fakat RTOS verilen görevleri yapmak için öncelik bazlı algoritmaları kullanmaktadır.
- GPOS iş parçacıklarını veya görevleri çalıştırırken sınırı olmayan bir sürede bunu yapabilmektedir. Bundan dolayı görevlerin yerine getirilirken bir gecikme meydan gelmektedir. RTOS, gecikme açısından değerlendirildiğinde GPOS göre daha az gecikme yaşanacak şekilde yapılacak görevleri gerçekleştirmektedir.
- RTOS genellik gömülü sistemlerde, hafif yapılı küçük cihazlarda çalışmaktadır. GPOS ise daha büyük sistemlerde çalışmak üzere kullanılmaktadır [10].

2.2.2. RTEMS

RTEMS(Real-Time Executive for Multiproccessor System) [12] bir gerçek zamanlı çalıştırıcıdır. RTEMS gömülü askeri uygulamalar için yüksek performanslı bir ortam oluşturması amacıyla geliştirildi. Bu uygulamalar aşağıdaki belirtilen özellikleri gösterir:

- Çok görevli uygulamaları çalıştırabilme
- Homojen ve heterojen çoklu mikroişlemci sistemleri
- Olaya dayalı, öncelik tabanlı, kesintli çizelgeleme
- Görevler arası iletişim ve senkronizasyon
- Dinamik bellek yönetimi
- Yüksek düzeyde kullanıcı yapılandırılabilirliği

RTEMS'i kullanarak programlama dilleri vasıtasıyla gerçek zamanlı uygulamalar geliştirilebilir. RTEMS çekirdeği programlama dilleri olarak Ada, C, Java desteği vermektedir. RTEMS projesi ilk olarak askeri uygulamalar için yazılmaya başladığında Ada programlama dili kullanılarak da gerçek zamanlı uygulamalar geliştirilmekteydi.

Teknolojinin gelişmesiyle beraber çok çekirdekli ve çok işlemcili sistemler oluşturulmaya başlandı. Bu tip sistemlerde çalışacak olan gerçek zamanlı uygulamalar Ada ile program yazıldığında çeşitli sorunlardan kaynaklı olarak istenilen performans elde edilemedi. Ada Programı ile yazılmış gerçek zamanlı uygumaların sorunları temel olarak şunlarıdır.

- Ada programlama dili mekanizma olarak çoklu görevi yerine getirmeye desteklemektedir fakat bu yetenekleri tek işlemcili sistemlere göre gösterememiştir.

- Ada derleyicileri modern füze sistemleri kullanılamaycak kadar yavaş ve yetersiz kalmıştır.

Aşağıdaki işlemci aileri ile uyumlu olarak çalıştırılabilir: [3]

- Adapteva Epiphany
- Altera NIOS II
- Analog Devices Blackfin
- Atmel AVR
- ARM
- Freescale (formerly Motorola) MC68xxx
- Freescale (formerly Motorola) MC683xx
- Freescale (formerly Motorola) ColdFire
- Intel i386 and above
- Lattice Semiconductor LM32
- NEC V850
- MIPS
- Moxie Processor
- OpenRISC
- PowerPC
- Renesas (formerly Hitachi) SuperH
- Renesas (formerly Hitachi) H8/300
- Renesas M32C
- SPARC v7, v8, and V9

Tez çalışmasında oluşturulan kıyaslama takımı Xubuntu üzerine kurulan RTEMS RTOS'unda test edildi. RTEMS kodları çalıştırılan simülatörde SPARC işlemcisi kullanıldı. Yazılan uygulamaların tümü RTEMS üzerinde çalıştırmak için gerekli değişiklikler yapılarak test edildi.

RTEMS ile yazılan uygulamalar yüksek seviyeli diller ile yazıldığında farklı bir işlemci mimarisi üzerine taşınırken daha az çaba sarfedilecektir.

2.2.3. Zephyr

Zephyr, birden fazla mimaride, kaynak kısıtlı aygıtlar için iyileştirilmiş, ölçeklenebilir, açık kaynaklı bir RTOS'dur. Linux, macOs ve Windows üzerinde kurularak çalıştıralabilir.

Zephyr projesini geliştirilme motivasyonu şu şekilde ifade edilebilir: nesnelerin internetinin (IoT) bir sonucu olarak oluşacak olan aygıtların geliştirilmesinde maliyetin düşürülmesinin ve hızlandırmanın yanında üretici firmalar arasında tarafsız bir bölge oluşturmak istenmesidir.

Zephyr RTOS 'u ile gömülü sistemlere sensörleri bağlayarak veri almak daha kolay hale gelecektir. Geliştiriciler ihtiyaç duydukları gerçek zamanlı sistem çözümleri için kullanabilirler.

Zephyr VxWorks RTOS 'u baz alınarak geliştirilmiştir. Sahip olduğu çekirdek DSP, RTOS teknolojisi üzerinden 20 yıla yakın bir sürede geliştirilmiştir. Zephyr RTOS'u resim işleme, telekomünikasyon, radar kontrol sistemleri, uydular vb. dahil olduğu çeşitli ticari uygulamalarda kullanılmaktadır. Zephyr RTOS'u Apache 2.0 lisansı altında geliştiricilerin ve araştırmacıların kullanımına sunulmaktadır.

Zephyr çekirdeği birden çok işlemci mimarisini tarafından desteklenmektedir. Desteklenen mimariler aşağıdaki şekilde listelenmiştir: [13]

- ARM Cortex-M,
- Intel x86,
- ARC, NIOS II,
- Tensilica Xtensa
- RISC-V 32 .

2.2.4. eCos

eCos (Embedded Configurable Operating System) açık kaynaklı bir RTOS'dur. Sağlandığı ücretsiz lisans ile geliştiriciler ve araştırmacılar kaynak kodları ücretsiz bir şekilde indirerek çalışmalarını yapabilirler. Ayrıca endüstride üretilmiş olan ürünler de kullanılmaktadır. eCos kullanarak üretimi gerçekleştirilmiş olan başarılı ürünlerden biride Brother HL-2400 lazer yazıcısı olmuştur [14].

eCos açık kaynaklı bir çalışma sistemini GNU araçları vasıtasıyla sağlar. Geliştiriciler çalışma zamanında bütün yönleri ile serbest olarak erişmek için RTOS'a müdahale edebilirler. Bu RTOS'un hiçbir özelliği değiştirilmemesi için kapatılmamıştır ve geliştiricilerin denemelerine açıktır. Geliştiricilere verilen bu düzenleme hakkı eCos lisansı tarafından sağlanmaktadır. eCos lisansı ile uygulamalar geliştirilebilir ve dağıtılabilir.

eCos'u çalıştırma ayarlarını değiştirmek için kullanılan ayarlar sistemi geliştiriciye gerçek zamanlı bileşenler üzerinde istediği gerekli değişiklikler yapmaya izin verir. eCos'un aksine geleneksel işletim sistemlerinde uygulama sahibinin gerçekleştirmek istedikleri sınırlandırılır. Geliştiriciler kendi özelliklerini belirledikleri bir işletim sistemi yapmayı sağlar ve gömülü sistemlerin çok çeşitli kullanım alanlarına uygun hale getirilebilir.

eCos ile herhangi bir telif ücreti ödemeden uygulamalarınızı değiştirebilir ve dağıtabilir. Ayrıca çalışma zamanı kaynak kodları ve eCos araçlarını ücretsiz olarak edinilebilir. Temel gömülü uygulamalar için her şeyi ücretsiz olarak geliştiricilerin kullanımına sunar. eCos'un geliştirilmesindeki temel motivasyon ucuz ve değiştirebilir bir RTOS'u daha doğrusu bir gömülü sistem ortamını oluşturmayı hedeflemektedir. Gömülü uygulama desteği için gerekli olan bellek yönetimi, istisna yönetimi, C, matematik kütüphaneleri vb. tüm işlevleri geliştiricilere sağlamaktadır.

eCos 16, 32 ve 64 bit mimariler dahil olmak üzere çok çeşitli hedeflenen mimarilere ve platformlarda çalıştırılacak halde tasarlanmıştır.

eCos 13 farklı mimari tarafından desteklenmektedir ve bu mimariler aşağıdaki gibi listelenebilir:

- 68K/ColdFire
- ARM (including ARM7TDMI, ARM9TDMI, Cortex-M, StrongARM, XScale)
- CalmRISC16 and CalmRISC32 (RedBoot only)
- Fujitsu FR-V
- Fujitsu FR30
- Hitachi H8/300
- Intel x86
- Matsushita AM3x
- MIPS
- NEC V8xx
- PowerPC
- SPARC
- SuperH

eCos'un dağıtılmak için Windows ve Linux versiyonları da vardır. Fedora, openSUSE ve Ubuntu dağıtımlarında x86 mimarisi için çalışması test edilmiştir. Windows versiyonlarında Microsoft Windows 2000 Professional, Windows XP ve Windows Vista altında test edilmiştir.

eCos hem bir grafik yapılandırma aracı hem de bir komut satırı aracılığı ile ayarları değiştirilebilir. eCos ile uygulama geliştirmek için en azından gcc derleyici, gdb hata ayıklayıcısı ve binutils araçlarına geliştirmek yapmak için gereklidir [15].

2.2.5. VxWorks

VxWorks gömülü işletim sistemleri için kullanması kabul görülmüş bir RTOS'dur. Wind River System firması tarafından geliştirilmiştir. Havacılık, uzay araçları yazılımları ve askeri uygulamalarda başarılı olarak kullanılmıştır.

VxWorks çok çekirdekli mimarilari destekleyecek şekilde geliştirilmiş gerçek zamanlı bir çalıştırıcı üzerinden yazılmıştır. Çoklu görev işlemleri paralel bir şekilde çoklu veri işleyerek yerine getirebilir [16].

VxWorks, IoT içindeki bağlı cihazlar için güvenilir bir çalış ortamı oluşturmaktadır. VxWorks ABB, Airbus, Alcatel-Lucent, BD Biosciences, Boeing, Delphi, Eurocopter, Huawei, Mitsubishi, NASA, Northrop Grumman, Siemens ve Varian gibi dünyaca ünlü firmalar IoT 'ye hazır ürünler oluşturmak için kullanmışlardır.

Vxworks 32, 64 bit ve çok çekirdekli ve işlemci mimarilerin geniş bir çoğunluğu tarafından desteklenmektedir. Ayrıca aşağıdaki işlemci mimarilerinde çalışması test edilmiştir: [18]

- ARM
- Intel
- Power Mimarisi

2.3. WCET

Gerçek zamanlı sistemlerde çalışma zamanın kritik olduğu uygulamalar mevcuttur. Gerçek zamanlı sistemlerin genel olarak katı ve yumuşak türleri vardır [5]. Özellikle katı gerçek zamanlı sistemler için bir uygulamanın WCET'inin bilinmesi önemli bir problemdir [18]. Endüstiride birçok çalışma alanında aktif olarak WCET'inin bilinmesi gerekir. Yumuşak gerçek zamanlı sistemlerde uygulanmanın beklenen zamanda çalışması çok önemli bir gereklilik değildir. Katı gerçek zamanlı sistemlerde gerçek zamanlı uygulamaların teslim zamanında çalışması önem arzetmektedir. Beklenen zamanda uygulamanın cevap verememesi telafisi mümkün olmayan maddi zararlara neden olabilir [19]. Örneğin bir hava yastığı sisteminin zamanında çalışmaması telafisi mümkün olmayan maddi hasarlara veya ölümcül sonuçlara neden olmaktadır [20]. Günümüzde Endüstri 4.0 teknolojik devrimi yaşanmaktadır. Üretilen cihazlar artık insandan ziyade otomasyon sistemleriyle fabrikalarda üretim yapılmaktadır. Bundan dolayı endüstride, zamanında işini yapamayan bir sistem üretim hatasına neden olabilir.

WCET'i belirlemek için yazılan uygulamaların sensörler aracılığı ile elde ettiği bilgileri göre ne kadar sürede tepki verebildiğini tespit etmek önem kazanmıştır. Bu problemi ortadan kaldırabilmek için gerçek zamanlı bir uygulamanın çalışma zamanını hesaplayabilen WCET analiz araçları oluşturulmuştur. Bu araçlar sayesinde programın çalışma zamanları gerçekte kullanılmadan bulunmaya çalışılır. Gerçek zamanlı sistemlerde net olarak analiz araçları sayesinde şu nokta çalışır dan ziyade WCET bulunmaya çalışılır. Endüstride kullanılan gerçek zamanlı sistemler de çalışan uygulamaların WCET'ini bulabilmek için WCET analiz araçları geliştirilmektedir.

Gerçek zamanlı bir işleminin son teslim zamanını belirlemek gerekir. Son teslim zamanı, çalıştırılan gerçek zamanlı uygulamaların WCET'ini belirleyebilmek için gerçek zamanlı işlemlerin parçası olan iletişim eylemlerinin önceden bilinmesi gerekir. Bir görevin WCET'i belirlendiğinde görevin başlaması ve bitmesi için üst bir sınır belirlenmiş olur. Belirlenen bu üst sınır olası tüm başlatma senaryoları için aşılamayacak şekilde kesin bir değer olmalıdır. Aşılamayacak konusu daha çok katı gerçek zamanlı sistemler için geçerli bir ifadedir [5].

Zamanlamanın önemli olduğu sistemlerde kritik görevlerin yerine getirilmesinin en geç ne zaman olacağını bulmak adına üst sınırları belirleyen çeşitli yöntemler geliştirildi. Bu yöntemlerin çoğunda analiz edilen programların çalışması sırasında geçen sürenin üst sınırı olarak kullanılabilecek WCET'i tahmin etmeyi amaçlamaktadır. Kullanıcıların girdi bilgisini sağlayabildiği koşullarda sistemin belirli durumlardaki çalışma durumuyla ilgili sonuçlar bulunabilmektedir [21].

WCET'i belirleyen araçların daha kapsamlı programlama yapılarını test edebilir hale gelmesi gerekmektedir. Bahsedilen program yapıları daha uzun ve karmaşık algoritmaları gerçekleştirecek yapıda olan çözümler için WCET tahminlerini yapan WCET analiz araçlarının daha doğru sonuç üretmesi gerekir.

WCET'i belirlemek için kullanılan üç adet teknik vardır.

- Uçtan uça ölçme
- Statik analiz
- Hibrit ölçüm tabanlı analiz [22]

Yukarıdaki bulunan teknikler alanlarına göre WCET analiz araçları geliştirilmiştir. Ücretsiz olarak kullanıma sunulan WCET analiz araçlarının kaynak kodlarına ulaşmak mümkündür. aiT [23] ve Bound-T [24] ticari ve ücretli yazılımlardır. Bunun yanı sıra OTAWA [25], Heptane [26], SWEET [27] ücretsiz olan WCET analiz araçlarıdır [22].

2.3.1. aiT

aiT gerçek zamanlı sistemlerde en kötü durum çalışma zamanını statik olarak hesaplayabilen bir WCET analiz aracıdır. WCET analiz yöntemi olarak statik analiz yöntemini kullanmaktadır. Bu WCET aracı aslında Airbus Fransa desteği ile tasarlanmıştır. Kritik havacılık elektroniği yazılımların zamanlama davranışını doğrulamak için kullanılmıştır [23].

aiT WCET aracını kullanan firmalar şunlardır: [23]

- AIRBUS
- DAIMLER
- Vestas
- OHB
- mtu
- NASA

2.3.2. Bound-T

Bound-T Tidorum Ltd tarafından geliştirilen ve belli bir noktaya kadar genişletilen bir WCET analiz aracıdır. Gömülü sistemlerde yığın kullanımı ve WCET sınırını hesaplamak için makine kodlarını statik analizi yaparak bulabilmektedir. Günümüzde Bound-T Tidorum Ltd tarafından geliştirilmesi durdurulmuştur. Devam edilmemesinin ticari ve teknik nedenleri vardır. Araç günümüzde açık kaynak kodlu ve ücretsizdir [24]. Kıyaslama takımı Intel 8051, ADSP-21020 ve SPARC V7 mimarileri üzerinde çalışabilmektedir [28].

2.3.3. OTAWA

OTAWA 2004 yılında geliştirilmeye başlanan bir WCET aracıdır. Geliştirilme amacı tamamen ücretsiz bir karşılaştırma aracı oluşturmaktır [30]. OTAWA (Open Tool for Adaptive WCET Analyses) WCET hesaplama ve makine kodlarında

programların statik analiz yapan C ile yazılmış bir kütüphanedir. OTAWA tamamen LPGL lisansı altında ücretsiz olarak ulaşılabilir. Fransa da Toulose üniversitesi IRIT laboratuvarlarında TRACES adlı takım tarafından geliştirildi. İkilik yapıdaki programlarda çalışarak WCET analizinde bulmada yardımcı olabilir. OTAWA PowerPc, ARM, Sparc ve M68HCS gibi birçok mimari tarafından başarıyla çalıştırılabilmektedir.

Bugüne kadar OTAWA nın kullanıldığı projeler şunlardır: [25]

- W-SEPT (french ANR)
- parMERASA (FP-7)
- MERASA (FP-7)
- MORE (french ANR)
- MASCOTTE (french ANR)
- R2D2 (ONERA internal project)

2.3.4. Chronos

Chronos C programlarının WCET tahminleri üretmek üzere oluşturulmuş bir statik analiz aracıdır. WCET tahminleri için statik kod analizi tekniğini kullanır. Chronos çalışması için C programı ikili formatta girdi olarak alması gerekir. WCET'i verilen C programının karmaşık modern bir işlemci üzerinde çalışması ile doğru sonuçlar üretebilir. Chronos araştırma topluluklarının ihtiyaç duyduğu açık kaynak dağıtımlı özel bir araçtır. Bu araçda da işlemci modelleri SimpleScalar mimari simülatörü tarafından yakalanmıştır. Chronos araştırmacılar için esnek, geliştirilebilir ve kolayca erişilebilirdir. Gömülü gerçek zamanlı uygulamalarda kullanılabilmektedir [30].

2.3.5. RapiTime

RapiTime ölçüm ve analizleri birleştirildiği WCET aracıdır. Program yapılarının analizi için programları küçük parçalara böler. Küçük parçaların çalışma zamanları hedef işlem için tam ölçülebilecek yapıdadır. Ölçüm zamanları birleştirerek WCET'i

yaklaşık olarak belirlenebilir. RapiTime WCET tahmin edebilir ve optimizasyon için en iyi yeri ayırt edebilir. RapiTime kritik gerçek zamanlı gömülü sistemlerin zamanlama performansını doğrulamak için otomotiv ve havacılık yazılım mühendisleri tarafından kullanılmıştır. RapiTime sistem performansını iyileştirilmesi ve WCET'in daha çabuk bulanabilmesi adına kullanılmaktadır [31]. RapiTime'nın çalışma yaklaşımı akademik alanda WCET çalışmalarına dayanmaktadır. RapiTime akademik alandaki bu prototipi pWCET olarak adlandırılmıştır [32] [33] [34].

2.3.6. Heptane

Heptane açık kaynak kodlu statik analiz yapan bir WCET analizi aracıdır. İlk versiyonu 2000 civarında tasarlandı. Birçok önbellek mimarisi için önbellek analiz yapabilen teknikleri bünyesinde bulundurmaktadır. MIPS ve ARM komut seti mimarisini destekler. Linux ve MacOS da çalıştırabilir. Windows üzerinde çalıştırılmak için destek vermemektedir. GPL lisansına sahip ve açık kaynak kodlu bir yazılımdır [35]. MIPS ve ARM v7 mimarileri üzerinde WCET tahminleri yapabilir [36].

2.3.7. SWEET

SWEET WCET hesaplama yöntemi olarak akış analizi yöntemini kullanır. WCET aracının ismi olan SWEET, SWEdish Execution Time Analysis Tool ifadesinin kısaltmasıdır. Bir araştırma takımı tarafından 2001 yılında İsveç de geliştirilmiştir. Ücretsiz olarak indirilip kullanılabilir [37].

BÖLÜM 3. KIYASLAMA

Günümüzde çeşitli disiplinlerde kullanılan kıyaslama takımları mevcuttur. Kıyaslama her türlü alanda problem olan bir konudur. Kıyaslama takımların ortak amacı performans belirlemektir. Bilgisayar bilimlerinde de bu problemi çözmek adına yazılmış kıyaslama uygulamaların sayısı bir hayli fazladır [22].

Bilgisayar bilimlerinde WCET'i tespit etmek amacıyla geliştirilen kıyaslama takımları vardır. Örneğin tam sayı ve kayan noktalı sayı hesaplamalar için DryStone kıyaslaması [38] ve sistem işlemcisi, hafıza alt sistemi ve derleyicinin etkisi için SPEC CPU2006 kıyaslaması vardır [39]. Bunlar endüstride ve akademik çalışmalar kullanılmaktadır. Kıyaslama takımlarının ücretli ve ücretsiz versiyonları vardır.

WCET ölçümü yapmak için kullanılan kıyaslama uygulamaları havacılık, otomobil, bilişim vb. alanlar da aktif olarak kullanılabilmektedir. Piyasada bulunan mevcut kıyaslama takımlarından bazıları şunlardır: EEEMBC, MiBench, DEBIE [40], PapaBench, Mälardalen vb.

Günümüzde mevcut olan kıyaslama uygulamalarında C, Ada ve Java programlama dillerini kullanarak yazılmıştır. JemBench java ile, TACLeBench C ile, Dhyrstone ise Ada programlama dili kullanılarak yazılmıştır. Dhyrstone kıyaslama takımı ilk olarak Ada ile yazılsada daha sonra C ile yazılan versiyonuda çıkmıştır [41].

Özellikler / Kıyaslama takımı	Ada	C	Java	Ücretli	Ücretsiz	RTOS desteği	Tek iş paraçıklı	Çok İş Parçacıklı
JemBench	-	-	+	-	+	-	+	+
Dhyrstone	+	+	-	-	+	-	-	-
EEMBC	-	+	-	-	+	-	+	-
Malardelen	-	+	-	-	+	-	+	-
MBBench	-	+	-	-	+	+	+	+
MiBench	-	+	-	-	+	-	+	-
PapaBench	-	+	-	-	+	+	+	-
PARSEC	-	+	-	-	+	-	+	+
PBench	-	+	-	-	+	+	+	+
TacleBench	-	+	-	-	+	+	+	+

Tablo 3.1. Kıyaslama takımları benzerlik ve farklıları tablosu

Oluşturmuş olduğumuz kıyaslama takımı MBBench ile mevcut olan diğer kıyaslama takımları ile karşılaştırma sonucu Tablo 3.1'de verilmiştir.

3.1. JemBench

JemBench kıyaslama takımı java programlama dili ile yazılmıştır. Klasik gömülü kontrol sistemlerinde kıyaslama yapmak için kulanılır. Kıyaslama uygulamaları mikro, çekirdek(kernel), uygulama, paralel, akış kategorilerine ayrılarak oluşturulmuştur. Uygulamalar minimum java standartını kullanarak çalışacak sekildedir. Kıyaslama çok çekirdekli ve çok iş parçacıklı sistemlerde çalışabilecek kıyaslama uygulamalarını içermektedir [42].

3.2. EEMBC

EEMBC sürüş, mobil resim işleme, nesnelerin interneti, mobil aygıtlar otonom sistemler de kullanılan yazılım ve donanım için performans kıyaslamalarında kullanılır. EEMBC gömülü işlemci uygulamalarının enerji verimliliği ve performans ölçümü için kullanılır. EEMBC 1997 yılında kurulan bir kurumdur ve kıyaslama sonuçlarını karşılaştırabilecek bir veri tabanı biriktirmektedir [43]. Kıyaslama takımı MIPS, POWERPC, ARM, PISA ve X86 mimarilerinden özellikleri toplandı [44].

3.3. MiBench

Gömülü kıyaslamaları kullanmak için yazılan MiBench araştırmacılar için ücretsiz olarak kullanabilir. EEMBC kıyaslama takımına benzer özellikleri vardır ve açık kaynak kodludur. MiBench altı kategoriden programlardan oluşmaktadır: otomotiv, endüstriyel kontrol, ağ, güvenlik, tüketici aygıtları, ofis otomasyonları, Telekomünikasyon. Gömülü işlemci performansı analizlerinde SPEC2000 göre daha başarılıdır [45]. Kıyaslama takımının uygulama kodları web sitesinde kullanıcıların kullanımına sunulmuştur [46].

3.4. PapaBench

PapaBench tamamı gerçek zamanlı yaşamda olan uygulamalardan oluşan bir kıyaslama takımı olarak tanıtıldı. İnsansız hava araçları için Paparazzi projesi temel alarak oluşturuldu. GNU lisansı ile lisanslanan bu kıyaslama takımı araştırmacıların kullanımı için ücretsiz olarak sunuluyor. Bu kıyaslama takımı WCET tahminlerinde ve çizelgeleme analizinde faydalıdır. Endüstride havacılık elektronik sistemleri üzerinde çalışan uygulamalar için WCET tahminlerinde daha gerçekçi tahminler yapılabilir [47].

3.5. PARSEC

PARSEC çok işlemcili sistemlerde çalışması için geliştirilmiş bir kıyaslama takımıdır. 9 adet uygulamadan ve 3 adet Çekirdek oluşmaktadır. Uygulamaların tamamı C/C++ ile yazılmıştır. Kıyaslamadaki her bir uygulamanın paralel çalışabilme özelliği vardır. Farklı işlemci mimarilerini desteklemektedir. Her bir çalışma alanı için farklı özelliklere sahip 6 adet giriş vektörü tanımlanmıştır [48].

3.6. PBench

Malardalen Kıyasalama takımında ki bir diğer eksikliği kapatmak üzere yazılmış olan kıyaslama takımlarından bir PBench dir. Paralel Takım kelimesinin kısaltmasıdır. PBench ‘e ve kaynak kodlarına internet medyası üzeriden ulaşılabilir [59]. PBench Paralel bir kıyaslama takımıdır ve bütün programlar tek çekirdekli olarak üretilmiştir. PBench 5 farklı uygulama içerir. Her uygulamanın hem tek çekirdekli hem de çok çekirdekli versiyonları vardır. Takımda toplam 10 adet program bulunmaktadır. Bütün programlar C programla dili kullanılarak yazılmıştır. PBench yazılma amacı uygulamaları hem paralel yapıları hem de ardışık olarak araştırmacılara test edebilme fırsatını vermektir [49].

3.7. Malardelen

Mälardalen kıyaslama takımı dünya çapında farklı araştırma takımları ve araç üreticileri tarafından toplandı. Kıyaslama uygulamaları C dili ile yazılmıştır. Her kıyaslama uygulamasının özellikleri ve kaynak kodları web sitesinde kullanıcıların kullanımına sunulmuştur [50]. Bu kıyaslama uygulamalarında yazılan tüm kıyaslama uygulamaları tek yollu çalışabilen programlardır [22].

3.8. TACLeBench

TACLeBench 53 adet kıyaslamanın oluşturduğu bir takımdır. Gömülü gerçek zamanlı sistemlerdeki kıyaslama ihtiyacını karşılamak için yazılmıştır. Standart kütüphane ve işletim sistemleri bulunmayan sistemlerde kullanımı kolaydır [52]. Kıyaslama takımının kodları github üzerinde paylaşılarak araştırmacıların kullanımına açılmıştır [52]. Kıyaslama Takımının adı TACLEBench Timing Analysis on Code-Level ifadesinin kısaltmasından oluşmaktadır [53].

BÖLÜM 4. MBBench

Oluşturduğumuz kıyaslama takımını MBBench olarak adlandırdık. 2018 yılında seçilen algoritmalar üzerinde uygulamalar toplandı. MBBench kıyaslama uygulamaları için yazılan uygulamalar linux işletim sistemi üzerinde başarılı bir şekilde çalıştırıldı. Uygulamalar RTEMS üzerinde çalıştırılacak hale getirilirken dikkat edilen en önemli nokta kodların değişmemesi olmuştur.

Kıyaslama takımının adı olan MBBench, Measurement Basemend Bench ifadesinin kısaltılmış halidir. Ölçüm tabanlı WCET analizi yapmak amacıyla oluşturulmuş bir kıyaslama takımıdır. MBBench uygulamaları dışarıdan girdi alarak çalışabilen uygulamalardan oluşmaktadır. Böylece çalışan her uygulama farklı patikalardan çalıştığı için çok yollu çalıştırılabilen uygulamalara sahiptir.

MBBench 30 adet programdan oluşmaktadır. Programların yarısı Xubuntu işletim sistemi diğer yarısı ise RTEMS çalışacak sekilde yazıldı. Yani 15 adet dışarıdan girdi vektörü alabilecek olan algoritmanların her biri için Linux ve RTEMS üzerinde çalışacak şekilde aynı kodlar değiştirilerek çalıştırıldı. Kıyaslama takımıyla yakalanmak istenen asıl hedeflerden biride gerçek zamanlı çalışacak kodları ile genel amaçlı işletim sistemi üzerinde çalışacak olan kodların çok değişiklik yapılmaması olmuştur.

MBBench uygulamaların 14 tanesi tek iş parçacıklı çalışacak şekilde yazılmıştır. Kalan 1 tanesi olan quicksort pogramı ise çok iş parçacıklı olarak çalışacak şekilde yazılmıştır. Merge_sort uygulaması sözde kod açısından birbirine benzeyen algoritmalardır. Bu iki algoritmanın merge_sort uygulaması tek iş parçacıklı olarak

yazılmıştır. İkiside herhangi bir tam sayılı diziyi sıralayan programlardır. Aynı dizi için sıralama işlemi yapıldığında çok iş parçacıklı çalışan uygulama olan quick sortun daha hızlı sıralama yaptığı gözlenmiştir.

MBBench uygulamalarının RTEMS üzerindeki oluşturulan kıyaslama takımı tek yollu çalıştırılabilme özelliği olan programlardır. Bunun nedeni komut satırı üzerinden veri alamamaktadır. Problemi çözmek için veri girdisi sağlamak için veriler kod içerisine gömülmüştür.

MBBench uygulamalarının RTEMS üzerindeki çalışan versiyonlarında girdi değeri üretmek için rastgele sayı üretme yolu kullanıldı. Bu şekilde çalıştırıldığında her seferinde aynı sayıların sistem tarafından üretildiği görüldü. Bundan dolayı program ne kadar çalıştırılırsa çalışsın sonuç olarak üretilen değerler aynı üretildiği gözlemlendi.

MBBench uygulamalarının Xubuntu üzerindeki oluşturulan kıyaslama takımı çok yollu çalıştırılabilen uygulamalardan oluşmaktadır. Her çalışmada komut satırından farklı girdi değerleri alabileceği için farklı patikaları takip ederek sonuç üretilecektir. Komut satırından almak yerine rastgele sayı üreterek uygulamalara farklı girdi değerleri sağlandığında da programlar çok yollu çalıştırılma özelliğini göstermektedir.

Kıyaslama takımı oluşturmak için Linux işletim sistemi olarak Xubuntu 17.10, gerçek zamanlı işletim sistemi olarak ise RTEMS tercih edildi. Yazılan kıyaslama uygulamaları için programlama dili olarak C tercih edildi. Kıyaslama takımını oluşturan algoritmalar dışarıdan test vektörü alacak şekilde çalışması öncesi belirlenen özellik matrisinide gözönünde bulundurarak 15 adet algoritma program yazmak için belirlendi.

MBBench'in kıyaslama takımı oluşturulurken gözönünde bulundurulan kriterler şunlardır:

- Programlama dili
- İşletim sistemi
- Algoritmalar

MBBench oluşturmak için izlediğimiz belli aşamalar vardır. Bir yazılım geliştirme projesi olduğu için 4 adet aşama sonunda kıyaslama takımı ortaya çıkmıştır. Bu aşamalar şunlardır: tasarım, geliştirme, test etme ve yayınlama.

4.1. Tasarım

MBBench kıyaslama takımını çalıştırmak için açık kaynak kodlu, Linux ailesinden Xubuntu işletim sistemini tercih ettik. Kıyaslama takımının yazıldığı dönemde en güncel Xubuntu sürümü 17.10'dur. Tezin yazım aşamasında Xubuntu işletim sisteminin en güncel versiyonu 18.10 olmuştur. Xubuntu işletim sisteminin en dikkat çeken yönü tamamen ubuntu altyapısını kullanırken Xfce masaüstü ortamıyla gerçek zamanlı işletim sistemleri için uygun bir çalışma ve performans ortamı yaratmaktadır. Böylelikle RTEMS üzerinde uygulamaların testi yapılırken ortaya çıkan sonuçlar Xubuntu varsayılan masaüstü ortamı Xfce'nin hafif kullanıcı arayüzü ortamı olmasından kaynaklı olarak linux ortamında benzer sonuçlar elde edileceği düşünüldü.

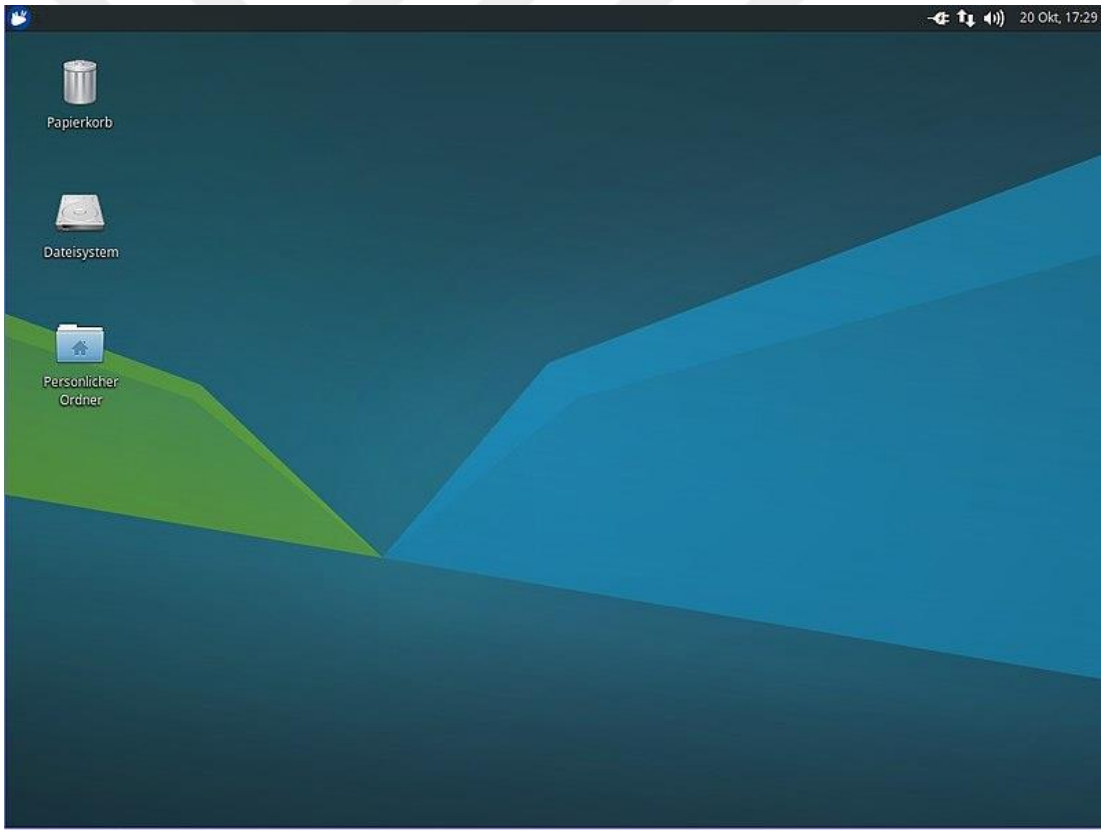
4.1.1. Xubuntu

Xubuntu, bir topluluk tarafından geliştirilen ubuntu tabanlı bir linux işletim sistemidir. Xubuntudaki "X", Xubuntu' daki masaüstü ortamı olan Xfce'nin kısaltmasıdır. Xfce, Unix işletim sistemleri için sunulan masaüstü ortamlarından biridir [55]. Görsel olarak şık olmasının yanında sistem kaynaklarını düşük ve hızlı kullanabilmektedir. Bunun haricinde kullanılan masaüstü ortamları Unity, Gnome,

KDE, LXDE dir. Sistem kaynakları düşük ve hızlı kullanması gerçek zamanlı yazılımları koşturmak için ideal bir işletim sistemi haline getirmektedir. Şekil 4.1. de Xubuntu masaüstünü çalışma görüntüsü verilmektedir.

Xubuntu işletim sisteminin ilk sürümü 2006 yılında yayımlandı. Açık kaynak kodlu bir işletim sistemidir. Canonical Ltd tarafından desteklenmektedir. GPL lisansının altında dağıtılan ücretsiz bir yazılımdır.

Ubuntu işletim sistemine benzerdir. Ubuntu işletim sisteminden farkı Unity masaüstü ortamı yerine Xfce masaüstü ortamı paketlenmiş olarak gelmektedir [55].



Şekil 4.1. Xubuntu 17.10 masaüstü ortamı

4.1.2. MBBench kıyaslama takımında bulunan uygulamalar

Algoritmaların sözde kodları üzerinden bakılarak herbir uygulama yazıldı. Çalışması ve sonuçlarıyla ilgili Linux ve RTEMS tarafında gerekli testler yapıldı. Algoritmalar da performans analizden ziyade belirlediğimiz özellik tablosundaki alanları desteklemesini dikkat ettik. Uygulamaların kaynak kodları, yazılan kodları açıklayan yorum satırlarınıda içermektedir. Programları çalışması sonucunda herhangi bir çıktı verilememektedir. Kıyaslama takımındaki kodlar saf yapıdadır. Yalnız test aşamasından sonuçlar görmek adına çıktı komutları kullanılmıştır.

MBBench kıyaslama takımı 30 adet programdan oluşmaktadır. Her bir algoritmanın iki adet uygulaması bulunmaktadır. Uygulamaların biri Xubuntu bir diğeri ise RTEMS uygulama kodlarını içermektedir. Tablo 4.1.'de MBBench programların algoritmaları ve programınların ne iş yaptıkları açıklanmıştır. MBBench'in programların gösteren Tablo 4.1. üç adet sütundan oluşmaktadır. Birinci sütunda programların isimleri verilmektedir. İkinci sütunda programın hangi algoritmayı kullanılarak yazıldığı belirtilmiştir. Üçüncü satırda ise programların ne iş yaptığı açıklanmıştır.

MBBench deki programlar farklı kategorilerden seçilmiş bilgisayar bilimine ait olan algoritmalardır. Algoritmalar şifreleme, sıralama, çarpma, sayı, karakter eşleme, sıkıştırma konularında kullanılanlardan seçildi. Seçim aşamasında dikkat edilen en önemli husus belirlemiş olduğumuz özellik matrisidir. Belirlediğimiz özellik matrisine uygun olanları kıyaslama takımımıza dahil ettik. Bazı uygulamalarda kodlar yazıldıktan sonra özellik matrisindeki bazı özellikleri uyması için kod üzerinde değişikler yaptık.

MBBench oluşturan uygulamaların algoritmalar 3 adet kaynaktan yararlanılarak belirlendi. Algoritmalarla ile ilgili daha fazla bilgi almak isteyen okuyucular

[56]'den yararlanabilirler. Booth çarpma algoritması ve sözde kodunu [57]'de bulunabilir. Cesar programının gerçekleştirildiği sezar algoritmasıyla ilgili [58]'den ayrıntılı bilgi alınabilir.

Program Adı	Algoritma	Açıklama
booth	Booth's algoritması	İki adet tek basamaklı pozitif sayı çarpar.
bucket_sort	Kova Sıralama algoritması	Tam sayılar sıralanır.
cesar	Sezar algoritması	Bir karakter katarını şifreler.
counting_sort	Sayma Sıralama algoritması	Tam sayılar sıralanır.
gcd	En büyük ortak bölen	İki pozitif sayının en büyük ortak bölenini bulur.
huffman	Huffman kod algoritması	Bir karakter katarını sıkıştırır.
knapsack	0-1 sırt çantası problemi	Bir sırt çantasına yerleştirebilecek maksimum değerini bulur.
merge_sort	Birleştirme sıralama algoritması	Tam sayılar sıralanır.
miller_rabin	Miller-Rabin asallık testi	Bir sayının asal olup olmadığını bulur.
pollard_rho	Pollard rho algoritması	Sayının çarpanını bulur.
quick_sort	Hızlı sıralama algoritması	Tam sayılar sıralanır.
rabin_karp	Rabin-Karp algoritması	Karakter katarı eşleme algoritmasıdır.
radix_sort	Taban sıralama algoritması	Tam sayılar sıralanır.
rsa	RSA kripto	Karakter katarı şifreleme ve şifre çözme.
standard_deviation	Standart sapma	Standart sapma hesaplanır.

Tablo 4.1.MBBench 1.0 Programlar

Kıyaslama takımındaki uygulamalardan ilki booth programıdır. Booth programı komut satırından alınan iki adet pozitif sayıyı çarpma işlemi yapmaktadır. Program içerisinde alınan iki sayıda ikili sayı düzenine çevirilirek işlem yapılır. Algoritmanın çalışma mantığı gereği çarpılan iki sayının her biri karşılaştırılan biti 0 ise sadece kaydırma işlemi yapılır. Eğer her karşılaştırma için 1 için çalışırsa çıkarma ve kaydırma işlemi yapılır.

Bucket_sort uygulaması kova sıralama algoritmasını gerçekleştirmektedir. Bu uygulamadaki ana amaç verilen tam sayılı diziyi sıralama işlemini gerçekleştirmektedir. Bizim uygulamamızda verilen n elemanlı diziyi belirlediğimiz formül gereği 3'e böler. Ardından her bir kovayı kendi içinde sıralar. En sonunda üç kova birleştiğinde karşımıza sıralanmış bir tam sayılı dizi çıkmaktadır.

Cesar uygulaması bir şifreleme algoritmasıdır. Verilen karakter katarı değerini belirlenen tam sayı değerine göre ötelenmektedir. Örneğin “ali” katarı için 1 rakamla öteleme işlemi gerçekleştiğinde şifrelenmiş metin “bmj” olacaktır. Şifrelenmiş metni çözmek için bilinmesi gereken tek şey ötelenme sayısıdır.

Counting_sort uygulaması 3 adet dizi üzerinden verilen tam sayı bir diziyi sıralayan bir programdır. Program girdi olarak verilen bir dizi, bu diziden farklı olarak sıralanmış olan dizinin yerini tutan ayrı bir dizi ve yazdırılacak dizi kullanılarak işlemler gerçekleştirilir. Bu uygulamada dikkat çeken en önemli durum dizilerin sıralanmış olan diziye kopyalanmadan önce B dizisi diye adlandırdığımız bir dizide sıralama yerleri oluşmaktadır. Sıralama işlemi gerçekleştirmek için 3 adet aynı boyutta dizi kullanma ihtiyacı ortaya çıkmıştır.

Gcd iki adet pozitif tam sayı için en büyük ortak bölenlerini veren bir uygulamadır. Bu uygulama için Öklid formülü lise ve orta okul düzeyindeki anlatılan şekildeki çözüme daha yakındır. Fakat Öklid formülü ile bulunan iki sayının ortak böleni uygulaması belirlediğimiz özellik tablosuna daha az uygun olduğu için tercih edilmemiştir.

Huffman programı verilen bir karakter katarını sıkıştırmak üzere yazılmış bir programdır. Verilen karakter katarındaki harfler frekansları ne kadar benzer olursa sıkıştırma oranı o aranda daha yüksek olacaktır. Tersine düşünüldüğünde ise verilen karakter bütün harfler farklı olduğunda sıkıştırma oranı daha az performanslı olacağı görülecektir. Uygulama verilen girdi değerine otomatik olarak harfleri kodlayacağı bit değerlerini belirleyebilmektedir.

Knapsack uygulaması bilgisayar bilimlerindeki popüler problemler arasındadır. Bu uygulamada ana amaç alınacak parça çantaya sığmıyorsa almamaktır. 0-1 sırt çantası problemine bir çözüm geliştirililmiştir. Bu aradaki 0 ve 1 eşya ya da parçayı

aldığında çanta tam kapasitisini aşıyormıyorsa işlem gerçekleştirilmektedir. Birim değeri yüksek olan çanta için en faydalı olan olacaktır.

Merge_sort uygulaması verilen bir tam sayılı diziyi sıralamak için kullanılır. Verilen dizi küçük parçalara bölünerek sıralama işlemi gerçekleştirilmektedir. Bu yönteme böl ve yönet tekniği denmektedir. C tarafından yazılırken görülen en önemli nokta dizinin sadece adı gönderilerek parçalanma işlemleri yapılmaktadır. Uygulama iyi incelendiğinde fonksiyona işlemesi için gönderilen dizin geri değer döndürmemesine rağmen dizi main fonksiyonu içerisinde sıralanmış bir biçimde alınabildiği görüldü.

Miller-rabin programa girdi olarak verilen bir sayının asal olup olmadığı tespit edebilmektedir. Uygulamanın Xubuntu tarafında sorunsuz olarak çalıştığı test edilerek kontrol edildi. Fakat RTEMS üzerinde çalışırken algoritma gereği içerisinde bulunan random fonksiyonu yüzünden Xubuntu tarafında alınan sonuçlar elde edilemedi.

Pollard_rho uygulaması verilen herhangi bir sayının çarpanını bulan bir programdır. Alınan herhangi bir sayı pollard_rho algoritmasına maruz bırakılarak çarpanları bulma işlemi gerçekleştirilir.

Quick_sort programı hızlı sıralama algoritmasının gerçekleşitiren bir uygulamadır. Girdi olarak alınan herhangi bir tam sayılı diziyi sıralama işlemini gerçekleştirir. Bu uygulamanın diğer uygulamalardan en önemli farkı kıyaslama takımı özellik tablomuzdaki yer alan özelliklerden çok iş parçacıklı özelliği destekleyecek şekil yazıldı. Yazılan uygulamanın RTEMS tarafından da Xubuntu tarafında başarı bir şekilde sıralama yapmayı gerçekleştirmiştir. Bunun yanı sıra RTEMS tarafında peş peşe çalıştırma sırasında dizi boyutları büyütüldüğünde çok iş parçacıklı çalışma modunda hatalar ortaya çıkmıştır. Merge_sort uygulamasına algoritması olan merge sort ile quick_sort uygulamasının algoritmaları yapı olarak birbirine benzemektedir.

Rabin_karp uygulaması verilen iki adet karakter katarını birbirine eş olup olmadığına karar veren bir uygulamadır. Rabin karp algoritmasını kullanarak sonuca ulaşmaktadır. Burada iki girdi eşit olarak çıkması için aralarında boşlukları dahi aynı olması gerekmektedir. Bu uygulamanın diğer bir özelliği ise kıyaslama takımındaki karakter katarı olarak girdi alabilme özelliği taşımaktadır.

Radix_sort uygulaması verilen bir tam sayılı diziyi sıralayabilen bir uygulamadır. Bu uygulama basamak değerleri kullanarak sıralama işlemi gerçekleştirilmektedir. Her aşamada basamak değerleri artıkça sıralama işlemi gerçekleşmektedir.

Rsa uygulaması alınan bir sayı girdi olarak verilen bir sayı ile şifreleyen ve bir şifrelenmiş sayı oluşturulmaktadır. Üretilen şifrelenmiş mesaj değeri oluşturulan anahtar kullanılarak tekrar açılabilmektedir.

Standard_deviation uygulaması girdi olarak alınan bir adet txt dosyası üzerinde okunan sayıların standart sapması bulunmaktadır. Linux tarafında çalışan uygulama başarılı br şekilde standart sapmayı hesaplayabilmektedir. RTEMS tarafında ise dışarıdan çeşitli yöntemler denenmesine rağmen dışarıdan txt ya da farklı bir formatta dosya okunamadığında standart sapma hesaplanamamıştır.

4.2. Geliştirme

İlk olarak uygulama yazılacak algoritmalar seçilmişti. Daha sonra algoritmalar yazmak için C programlama dili tercih edildi. Bunun yanında Java ya da Ada gibi programlama dilleri de tercih edilebilirdi. Biz gerçek zamanlı işletim sistemlerin geliştirilmesinde yaygın olarak kullanılan C dilinde yana tercihimizi kullandık. MBBench'teki yazılan uygulamaların tamamı C dili kullanılarak yazılmıştır.

C programlama dili gcc derleyicisi ile derlenip çalıştırabilecek özellikte bir yapıya sahiptir. Gerçek zamanlı uygulamalar Ada ve C dili ile de yazılabilmektedir.

Programlama dili tercihi yapıldıktan sonra kıyaslama takımının uygulama kodlarının çalıştırılacağı işletim sistemleri belirlenmiştir. Bu nokta iki adet işletim sistemine destek verilmeye karar verildi. Linux dağıtımları için Xubuntu ve gerçek zamanlı sistemler için RTEMS tercih edildi. Bu seçimin iki adet nedeni vardır. Birincisi bu iki işletim sisteminin hem endüstri hem de akamedemik alanda kullanılmaktadır. İkincisi ise RTEMS yıllardan beri gerçek zamanlı sistem geliştirilmesinde kullanıldığında oluşabilecek hatalar ve çözümleri hakkında daha geniş bir destek ortamı bulanabilir. Ayrıca gömülü sistemler için yazılmış olan kıyaslama uygulamaları incelendi. Bu uygulamalar inceledeğinde RTEMS geliştiricilerinin gömülü sistem yazma konusunda tercih edilmektedir. Fakat RTEMS'i baz alan kıyaslama takımları sayısı fazla değildir.

Her iki işletim sisteminde yazılacak versiyonlar belirlendi. Daha sonra belirlenen bu versiyonlara göre her programın hem Xubuntu ve hem de RTEMS üzerinde yazdık. Farklı platformlarda yazılan programların daha iyi karşılaştırılması için aynı programların farklı versiyonlarını mümkün olduğunca benzer şekilde yazmaya çalıştık.

Son olarak kıyaslama uygulamalarının oluştuğunda taşıması gereken kriterlerini belirledik. Belirlenen kriterler aşağıdaki şekilde listelenmiştir:

- Kıyaslama takımını oluşturak için ana motivasyonumuz ölçüm bazlı WCET analizidir. Bu nedenle, girdi alan ve bu girdiler üzerinde çalışan programları seçmeye ihitiyacımız vardır.

- Her program çeşitli program yapıları (döngüler, kararlar vb.) içermesi gerekir.
- Her program belirlenen özellik matrisindeki birden fazlasını içerebilir (Tek iş parcaçıklı, harici rutin kullanımı vb.).
- Programlar farklı veri tipleri (kayan nokta, tam sayı vb.) ve veri yapıları (dizi) üzerinde işlemler yapmalıdır.
- En az bir program bit seviyesi işlemleri içermelidir.

Kıyaslama uygulamalarında bazıları girdi olarak tek veri alırken bazıları ise bir test vektörü (girdiler dizisi) almıştır. Belirlenen özelliklere göre programlar ve özelliklerini gösteren bir matris hazırlandı. Bu matris ve programların taşıdıkları özellikler Tablo 4.2.'i üzerinde verildi.

Kıyaslama takımı yazılırken girdi alan algoritmalar seçilmeye çalışıldı. Algoritma seçilerken bilgisayar biliminde kullanılan çeşitli alanlardan seçildi. Her yazılan algoritmadan sonra girdi alınacak özelliğe göre bir sonraki algoritma seçimi yapıldı.

Test vektörü alan kıyaslama takımı github üzerindeki oluşturulan bir depoda web ortamında isteyen kişilerin kullanımına sunulmaktadır [59].

Gereksinimlere bağlı olarak, desteklemek istediğimiz özellikleri belirledik. Bundan sonra basitlik için her özelliğe bir kısaltma verildi. Özellikler Tablo 4.2.'de gösterilmiştir.

Ad	Kısaltma	Açıklama
Tek İş Parçacıklı	ST	Program tek iş parçacıklıdır.
Çok İş Parçacıklı	MT	Program çok iş parçacıklıdır.
Dışarıdan rutin	ER	Program harici rutin kullanır.
Tek yollu	SP	Program her çalıştırmada aynı çalıştırma yolunu çalıştırır.
Çok Yollu	MP	Program her çalıştırmada farklı çalıştırma yollarını kullanır.
Dinamik Bellek Ayırma	DM	Program dinamik hafıza tahsisi yapar.
Döngü	L	Program döngüler içerir.
İçice Döngü	NL	Program içice döngüler içerir.
Özyineleme	R	Özyinelemeli fonksiyon çağrıları mevcuttur.
Karar	D	Karar yapıları (if…else vb.) kullanılır.
Dizi	A	Program dizileri üzerinde çalışır.
Bit Düzey İşlem	BLO	Bit düzey işlemler mevcuttur.
Kayan Noktalı İşlem	FPO	Kayan noktalı işlemler mevcuttur.
Tam Sayılı İşlem	IO	Kayan noktalı işlemler mevcuttur.
Girdi Vektörü	IVEC	Program girdi olarak bir vektör alır.
Girdi Verisi	IVAL	Program girdi olarak tek bir değer alır.
Girdi Dosyası	IF	Program girdi olarak bir dosya alır.

Tablo 4.2.Özellikler

Yukarıda belirtilen özellikler çerçevesinde tüm program yapılarını ve özelliklerini başarıyla temsil eden bir kıyaslama takımı oluşturmaya çalışıldı.

Belirtilen özellikler göre başlayan algoritma yazım çalışmaları sonucunda test vektörü alabilen kıyaslama uygulama takımı olan MBBench'i oluşturarak çalışmalar tamamlandı.

Yukarıdaki göz önüne alınan özelliklere göre algoritmaların uygulama kodlarında gerekli değişikler yapıldı. Linux de çalışan uygulamaların tamamı komut satırı üzerinde veri alacak şekilde tasarlandı. Ve bu alınan veriler kullanılarak programın farklı sonuçlar elde edilebilmektedir.

MBBench'deki programların linux versiyonun taşıdığı özellikler Tablo 4.3.'de gösterilmiştir. Tablodaki bir sütün ile satırın birleştiği yerde (+) ve (-) işaretleri bulunmaktadır. Tablodaki dikey sütünlar program adlarını taşımaktadır. Programdaki

düşey sütunlar ise kıyaslama takımının bir özelliğini ifade etmetedir. Bu hücrelerdeki gösterilen her bir işareti programın o hizadaki özelliği taşıdığını göstermektedir. Eğer bir hücrede (+) işareti var ise program tablonun o satırında bulunan özelliği taşımaktadır manasına gelir. Diğer bir yandan eğer bir hücrede (-) işareti taşıyor ise programın o satırda bulunan özelliği taşımadığını gösterir.

Programlar / Özellikler	booth	bucket_sort	cesar	counting_sort	gcd	huffman	knapsack	merge_sort	miller_rabin	pollard_rho	quick_sort	rabin_karp	radix_sort	rsa	standard_deviation
ST	+	+	+	+	+	+	+	+	+	+	-	+	+	+	+
MT	-	-	-	-	-	-	-	-	-	-	+	-	-	-	-
ER	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
SP	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MP	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
DM	-	-	-	-	+	-	-	-	-	-	+	-	+	-	-
L	+	+	+	+	-	+	+	+	+	+	+	+	+	+	+
NL	-	+	-	+	+	+	+	-	-	-	-	+	+	-	+
R	-	-	-	-	-	-	-	+	+	+	+	+	+	+	+
D	+	+	+	+	+	+	+	+	+	-	+	+	+	+	+
A	+	+	+	+	-	+	+	+	-	-	+	+	+	-	+
BLO	+	-	-	-	-	-	-	-	-	-	-	-	-	-	-
FPO	-	-	-	-	-	+	-	-	-	-	-	-	-	+	-
IO	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
IVEC	+	+	+	+	+	+	+	+	-	-	+	+	+	+	-
IVAL	-	-	-	-	-	-	-	-	+	+	-	-	-	-	-
IF	-	-	-	-	-	-	-	-	-	-	-	-	-	-	+

Tablo 4.3. MBBench 1.0 Programları (Linux versiyonu) özellik matrisi

MBBench programların Linux versiyonu olarak verilen Tablo 4.3. incelenirse herhangi bir programın bütün özellikleri taşımadığı görülebilir.

MBBench deki programların RTEMS versiyonun taşıdığı özellikler Tablo 4.4.'de gösterilmiştir. Tablodaki bir sütün ile satırın birleştiği yerde (+) ve (-) işaretleri bulunmaktadır. Linux versiyonu gösteren Tablo 4.3.'deki belirlenen özelliği taşıyan

program bulunduğu hücreye (+) işareti konulmuştur. Özelliği taşımayan programların hizasına gelen hücrede ise (-) işaretleri konulmuştur.

RTEMS versiyonu için düzelenen Tablo 4.4.'e göre bütün programların tek yollu çalıştığı çıkarımı yapılabilir. Çünkü girdi değerleri programın içine gömüldüğü için programlar her çalışmada aynı yolu takip ederek çalışacaktır. Linux versiyonuna göre bir farkda girdi dosyası özelliğinden göze çarpmaktadır. RTEMS versiyonundada girdi dosyası alarak program çıktı verilememiştir. Aynı zamanda RTEMS versiyonu için girdi vektörü, girdi değeri özellikleride karşılanamamıştır.

Programlar / Özellikler	booth	bucket_sort	cesar	counting_sort	gcd	huffman	knapsack	merge_sort	miller_rabin	pollard_rho	quick_sort	rabin_karp	radix_sort	rsa	standard_deviation
ST	+	+	+	+	+	+	+	+	+	+	-	+	+	+	+
MT	-	-	-	-	-	-	-	-	-	-	+	-	-	-	-
ER	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
SP	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
MP	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DM	-	-	-	-	+	-	-	-	-	-	+	-	+	-	-
L	+	+	+	+	-	+	+	+	+	+	+	+	+	+	+
NL	-	+	-	+	+	+	+	-	-	-	-	+	+	-	+
R	-	-	-	-	-	-	-	+	+	+	+	+	+	+	+
D	+	+	+	+	+	+	+	+	+	-	+	+	+	+	+
A	+	+	+	+	-	+	+	+	-	-	+	+	+	-	+
BLO	+	-	-	-	-	-	-	-	-	-	-	-	-	-	-
FPO	-	-	-	-	-	+	-	-	-	-	-	-	-	+	-
IO	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
IVEC	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
IVAL	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
IF	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Tablo 4.4. MBBench 1.0 Programları (RTEMS versiyonu) özellik matrisi

RTEMS gerçek zamanlı bir gömülü işletim sistemi olduğundan, programları komut satırından çalıştırmak ve böylece komut satırı ortamından girdi almak için kıyaslama takımı elde edilememiştir. Önemli bir not olarak şunu belirtmek gerekirse RTEMS

sürümleri komut satırından girdileri desteklememektedir. Bölüm 5.1.'de daha ayrıntılı bir açıklama yazılmıştır.

Programların Linux versiyonlarının derlenmesi GNU Compiler Collection (GCC) 7.2.0 kullanılarak yapılmıştır. Hem derleme hem de test çalışmaları Xubuntu 17.10 linux dağıtımında gerçekleştirildi.

Programların RTEMS versiyonları, RTEMS 5.0.0 ile versiyonu ile derlenmiştir. Test işemleri SPARC simülatöründe gerçekleştirildi.

Yukarıdaki kıyaslama takımı için verilen özellikler tablosunu incelenebilir. Uygulamaların tümü tek çekirdekli olarak çalışabilir durumdadır. MBBench 1.0 sıfırda çok çekirdekli çalışan bir uygulama quicksort haricinde mevcut değildir.

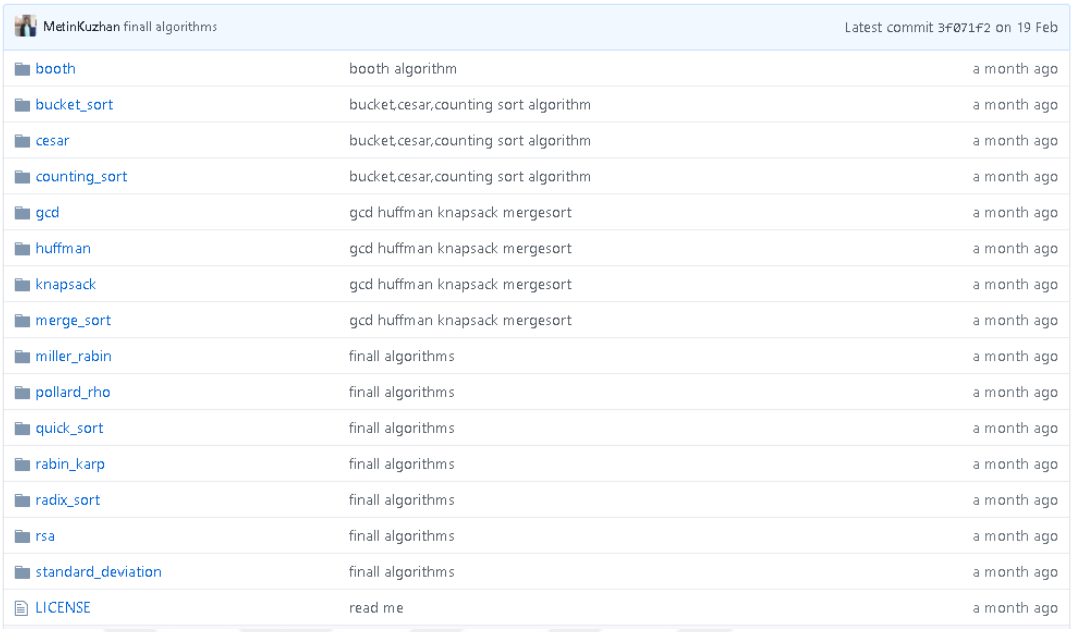
Standart sapma uygulamasını girdi olarak harici bir dosya üzerinden elde etmektedir. RTEMS üzerinde çalışan uygulama da harici dosya üzerinde girdi alma işlemi yapılamamıştır.

Booth kıyaslama uygulaması bit düzeyinde işlemi gerçekleştirebilir. MBBench 1.0'deki en uzun kod satırına sahip uygulamadır.

4.3. Detaylar

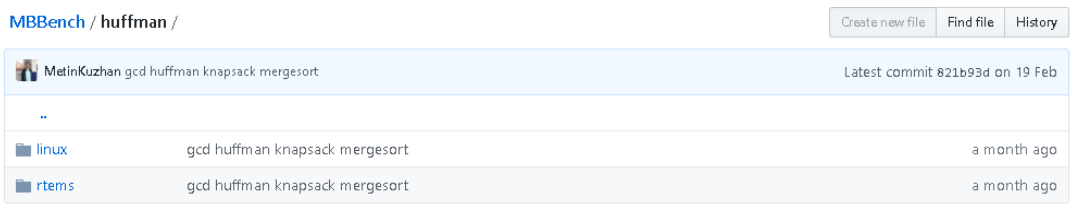
4.3.1. Kıyaslamanın takımında bulunan uygulamaların dizin yapısı

Kıyaslama takımı web üzerinde depolanmış şekilde ilgilen bilim insanlarının kullanımına açıldı. Şekil 4.2.'de MBBench WCET kıyaslama takımının klasör görünümü verilmiştir.

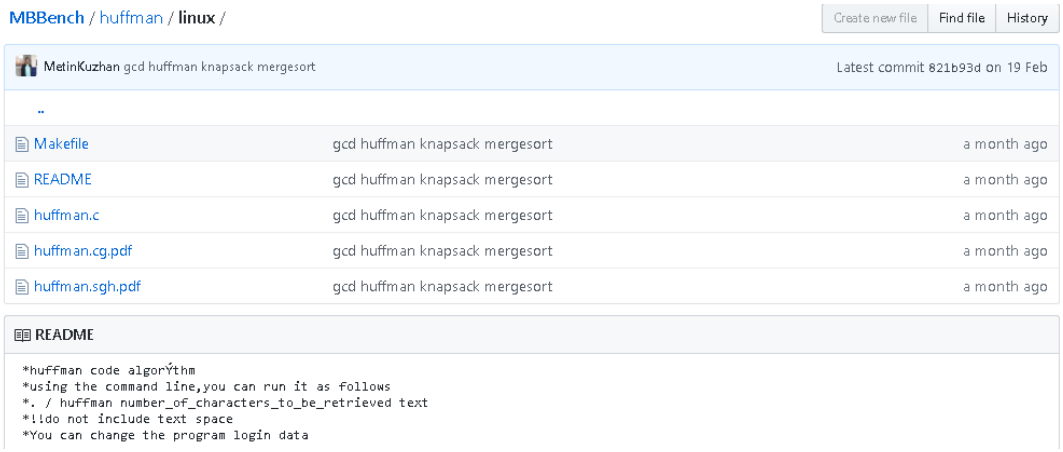


Şekil 4.2. MBBench uygulaması klasör yapısı

Her bir uygulama için ayrı birer dizin oluşturuldu. Dizinin içerisinde Linux ve rtems adında iki adet dizin bulunmaktadır. Her bir dizin kendi içerisinde uygulamanın c kodu, çağırma grafiği, kapsam hiyearşi grafiği, makefile dosyası ve çalıştırmak için gerekli bilgilerin yazıldığı dosyaya sahiptir. Huffman programı için Şekil 4.3.'de huffman uygulamasının klasör yapısı gösterilmiştir.

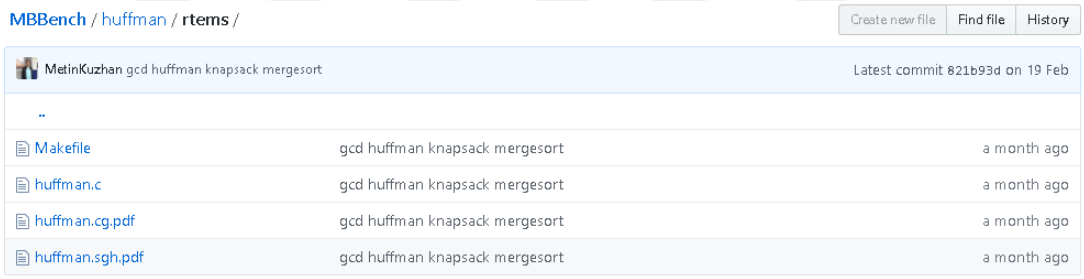


Şekil 4.3. Huffman uygulaması klasör yapısı



Şekil 4.4. Huffman uygulamasının Linux'un klasör yapısı

Örnek olarak verilen huffman uygulamasındaki linux dizinin klasör görünümü Şekil 4.4.'de verildi. README dosyasına bakılarak huffman uygulaması Xubuntu terminal ekranı üzerinden çalıştırabilir.

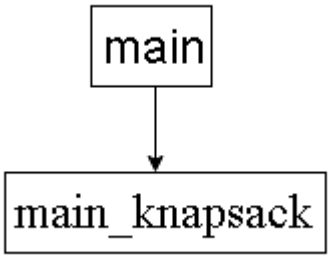


Şekil 4.5. Huffman uygulamasının RTEMS'in klasör yapısı

Örnek olarak verilen huffman uygulamasındai RTEMS dizinin klasör görünümü Şekil 4.5.'de verildi. README dosyasına bakılarak huffman uygulaması RTEMS üzerinde çalıştırabilir.

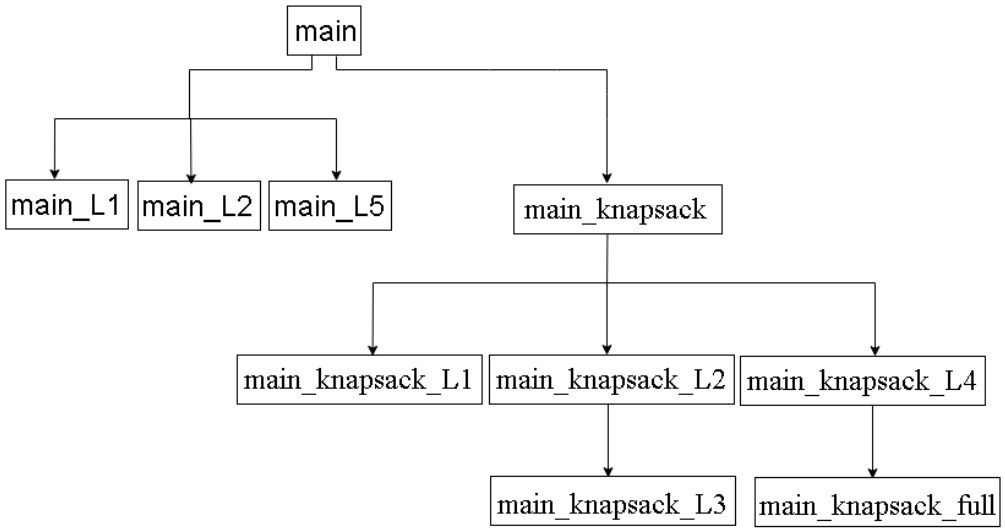
Verilen makefile dosyaları ile programlar komut satırı üzerinde derlenebilirler. Programları çalıştırmadan önce özellik tablosuna kullanım ihtiyacına göre bakmak faydalı olacaktır. Her bir programı çalıştırmadan önce program ile ilgili README dosyasını okumak doğru bir şekilde çalıştırmak ve sonuçlar almak için önemlidir.

Çağırma ve kapsam hiyerarşi grafiği bütün program linux dizinlerinde mevcuttur. Örnek olarak knapsack.c kıyaslama uygulamasının grafikleri verildi. Şekil 4.6.'da knapsack programının çağrı grafiği Linux versiyonu için verilmiştir. Çağrı grafiği incelediğinden programın main ve knapsack fonksiyonları üzerinde çalışarak sonuç üretilmektedir.



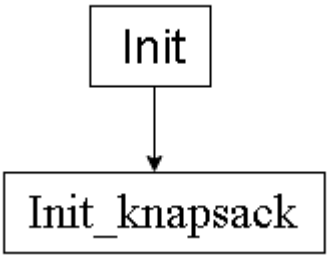
Şekil 4.6. Knapsack Programının çağrı grafiği (Linux versiyonu)

Şekil 4.7.'de knapsack programının linux versiyonu için kapsam hiyearşi grafiği verildi. Grafik incelediğinde Programın 6 adet döngü içerdeği görülmektedir. Ayrıca 2 adet de içiçe döngü içermektedir.



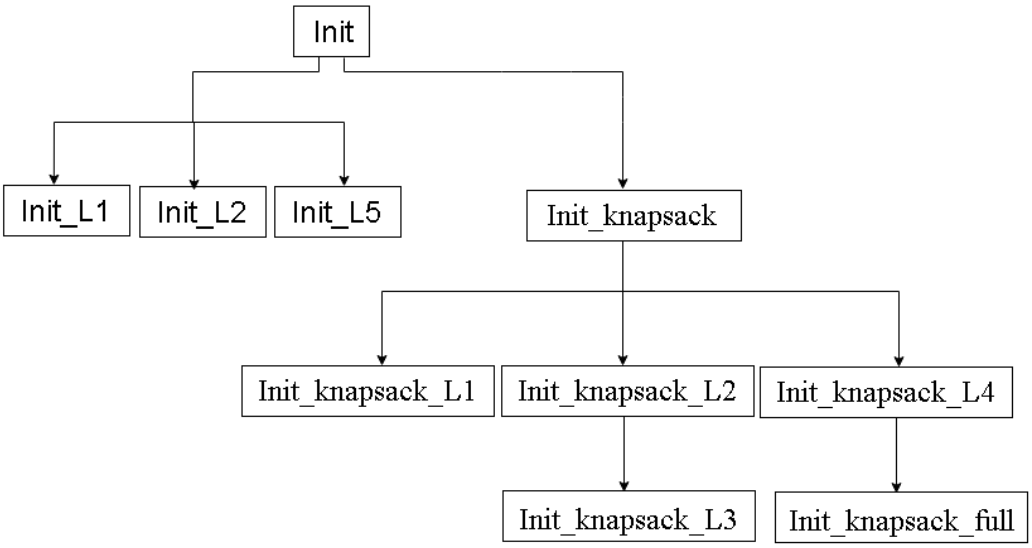
Şekil 4.7. Knapsack programının kapsam hiyearşi grafiği (Linux versiyonu)

Çağırma ve kapsam hiyerarşi grafiği bütün programların rtems dizinlerinde mevcuttur. Örnek olarak knapsack.c kıyaslama uygulamasının grafikleri verildi. Şekil 4.8.'de knapsack programının çağrı grafiği RTEMS versiyonu için verilmiştir. Çağrı grafiği incelediğinden programın Init ve knapsack fonksiyonları üzerinde çalışarak sonuç üretilmektedir.



Şekil 4.8. Knapsack Programının çağrı grafiği (RTEMS versiyonu)

Şekil 4.9.'da knapsack programının rtems versiyonu için kapsam hiyearşi grafiği verildi. Grafik incelediğinde Programın 6 adet döngü içerdeği görülmektedir. Ayrıca 2 adet de içiçe döngü içermektedir.

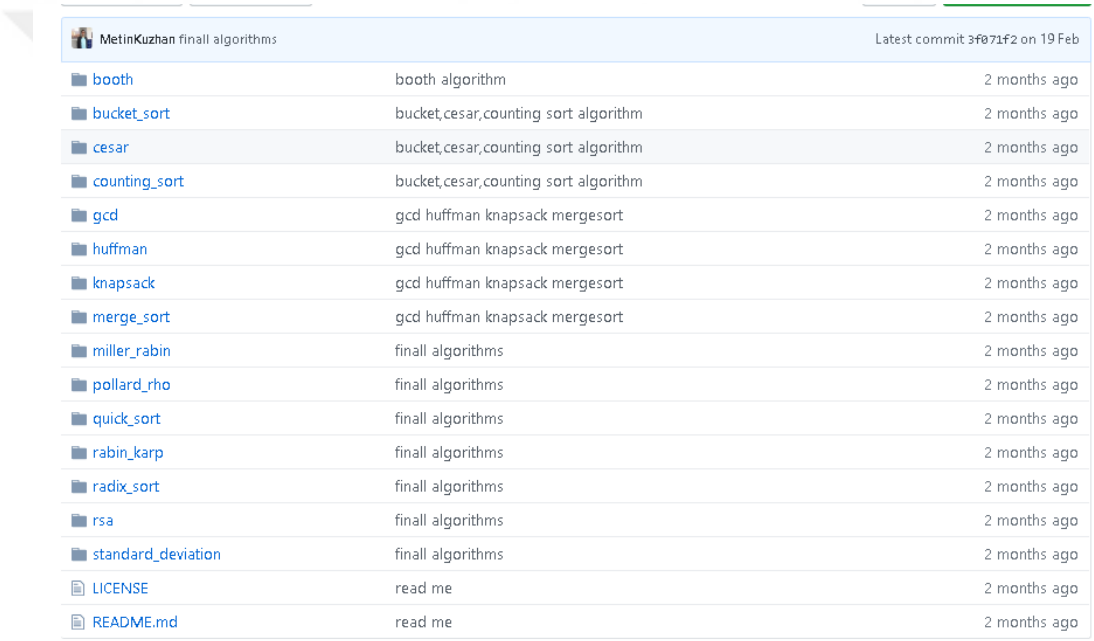


Şekil 4.9. Knapsack programının kapsam hiyearşi grafiği (RTEMS versiyonu)

Her iki grafik inceldeğinde kodların sadece ana çalıştırma fonksiyonu linuxde main, rtems versiyonunda ise Init olduğu görülmektedir. Bunun haricinde kodlarda herhangi bir değişiklik yapılmamıştır.

4.4. Yayımlama yeri

MBBench 1.0 kıyaslama takımı web üzerinde bilim insanlarının kullanımına açıldı. Yayımlanan depo üzerinde kıyaslama uygulamalarına ve kaynak kodlarına ulaşılabilmektedir. MBBench 1.0 kıyaslama takımının Github deposundaki görünümü Şekil 4.10.'da gösterilmiştir.



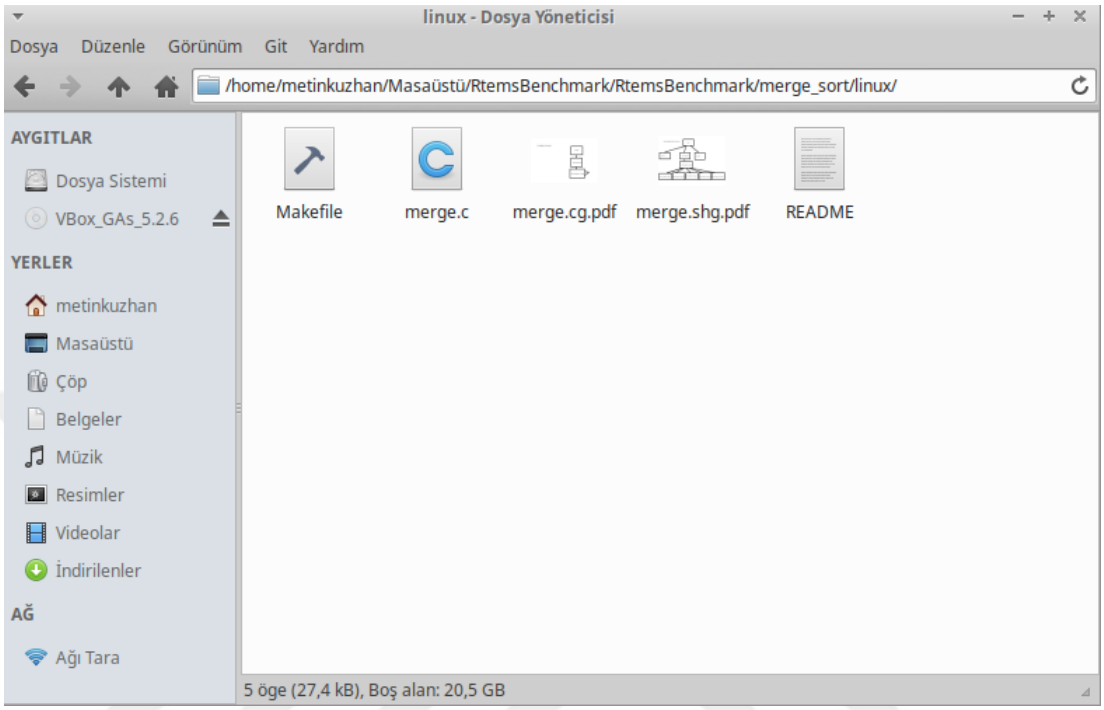
Şekil 4.10. MBBench 1.0 Github deposu

4.5. Test etme

4.5.1. MBBench teki bir uygulamanın Xubuntu'da çalıştırılması

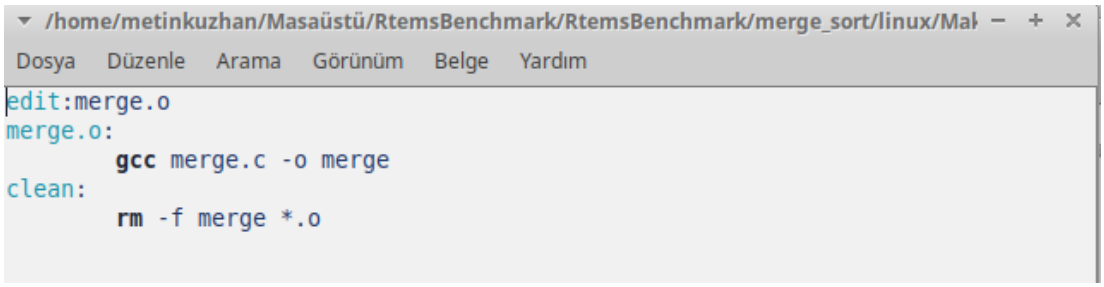
Oluşturduğumuz kıyaslama takımındaki herhangi bir uygulama Xubuntu terminal ekranı üzerinden çalıştırılabilir durumdadır. Çalıştırılmadan paylaşmış olduğumuz makefile dosya üzerinden uygulama derlenebilir. Derleme işleminden sonra programa dışarıdan veri sağlamak için terminal ekranından çalıştırılırken girdiler

yazılmalıdır. Merge_sort uygulmasının derlenip çalıştırılmasının adım adım aşağıdaki şekildedir. Merge_sort uygulamasının Linux üzerindeki uygulama kodlarının bulunduğu klasör Şekil 4.11.'de gösterilmiştir.



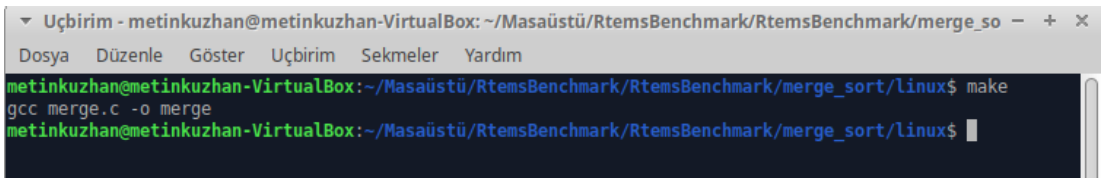
Şekil 4.11. Merge_sort klasörü görünümü

Terminal ekranı merge.c ve makefile klasörünün bulunduğu yerde açılır. Merge_sort uygulamasının makefile dosyası Şekil 4.12.'de gösterilmiştir.



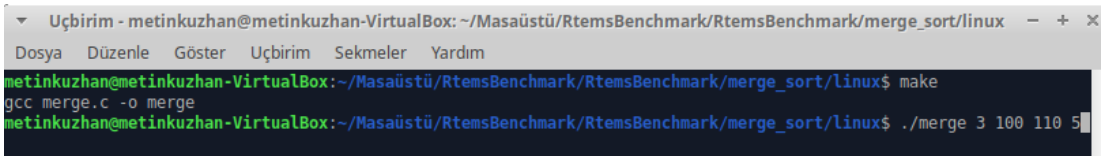
Şekil 4.12. Linux için merge.c' nin makefile dosyası

Terminal ekranı içerisinde make komutu verilerek merge.c dosyası derlenir. Derleme işlemi Şekil 4.13.'de gösterilmiştir.



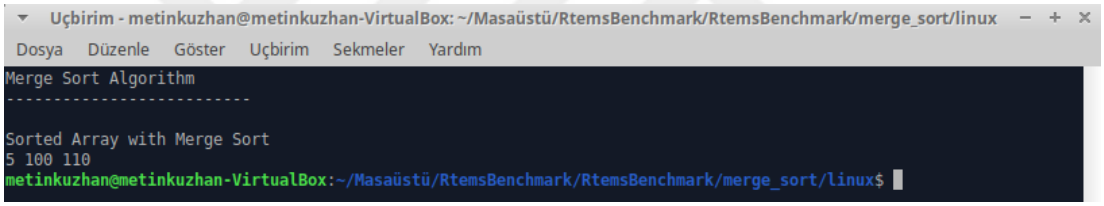
Şekil 4.13. Linux için merge.c derlenmesi

Derlenmenin ardından çalıştırmak için Şekil 4.14.'de gösterildiği gibi merge uygulaması çalışması için gerekli veriler girilir. Programın çalışma şekli programın dizini içerisinde bulunan README dosyası bakılarak yazılabilir. Bu uygulama için örnek kullanım şu şekildedir: /merge Adet Adet1 Adetn. Kullanım şekline bakılarak sıralanacak sayı adeti ilk parametre olarak girilir. Sonrasında sayılan adet kadar komut satırına parametre olarak sayı girilir.



Şekil 4.14. Linux için merge uygulamasının çalıştırılması

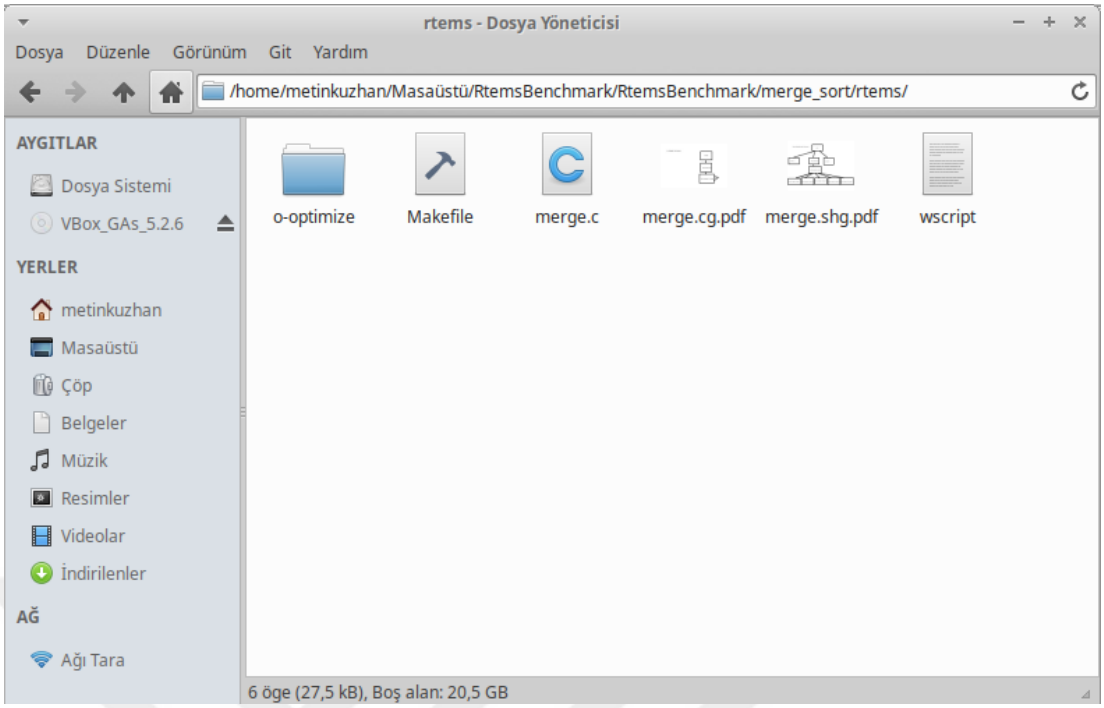
Programın çalışması sonucunda sıralanmış dizi yazdıralacaktır. Merge_sort uygulamasının Xubuntu işletim sistemin üzerinden çalışma sonucu Şekil 4.15.'de gösterilmiştir.



Şekil 4.15. Linux için merge uygulamasının çalışması oluşan sıralanmış dizi

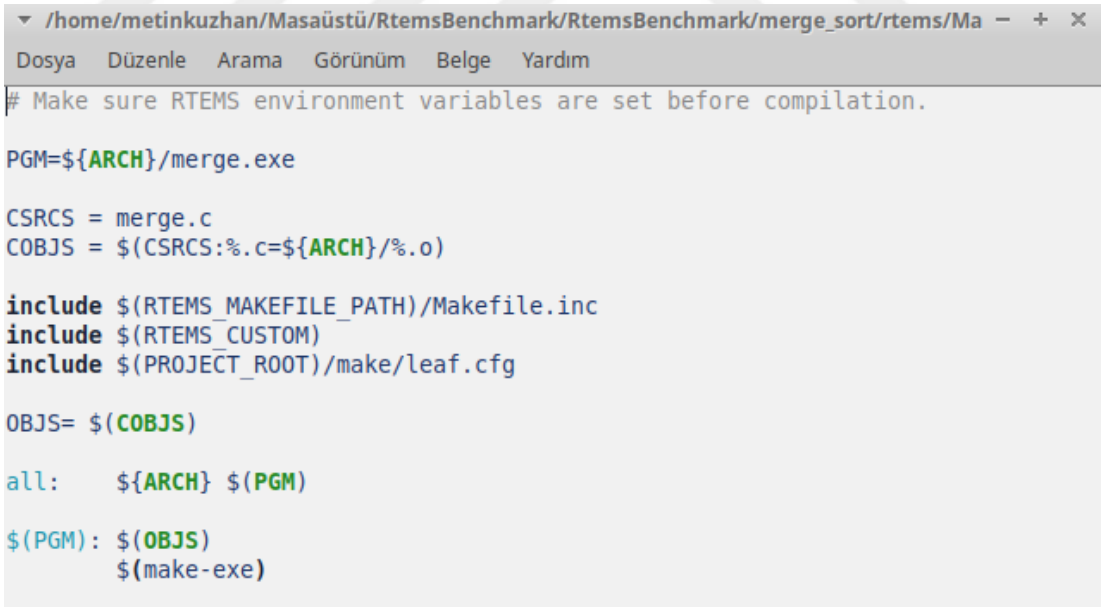
4.5.2. MBBench teki bir uygulamanın RTEMS'de çalıştırılması

Oluşturduğumuz kıyaslama takımındaki herhangi bir uygulama RTEMS'de Xubuntu'nun terminal ekranı üzerinden çalıştırılabilir durumdadır. Çalıştırılmadan paylaşmış olduğumuz makefile dosya üzerinden uygulama derlenebilir. Derleme işleminden sonra programa dışarıdan veri sağlamak için terminal ekranından çalıştırılırken girdiler yazılmamaktadır. Bunun yerine program girdi sağlamak için uygulama içerisine veri girilmiştir. Eğer girdi verilerini değiştirme istenirse kaynak kod içerisinde değişikler yapılabilir. Merge_sort uygulmasının derlenip çalıştırılmasının adım adım aşağıdaki şekildedir. Şekil 4.16.'da merge_sort uygulamasının rtems klasörünün yapısı gösterilmektedir.



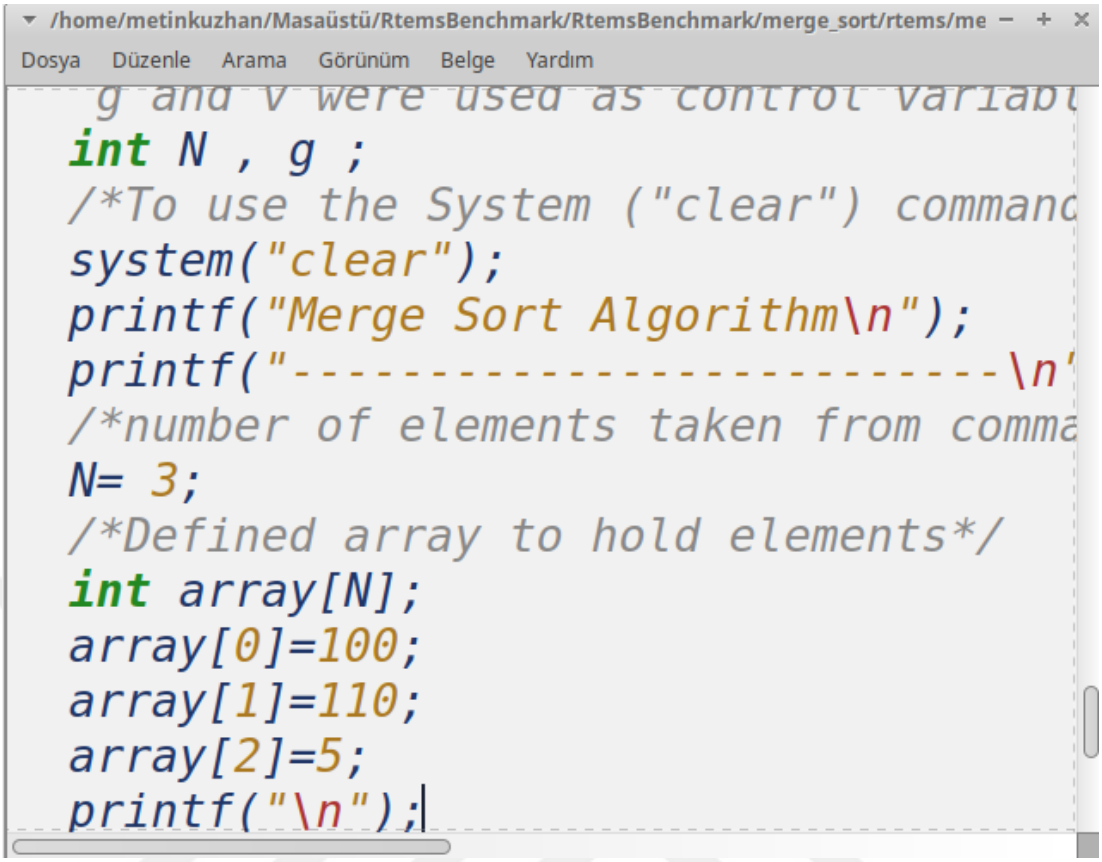
Şekil 4.16. RTEMS için merge_sort klasörü görünümü

Terminal ekranı merge.c ve makefile klasörünün bulunduğu yerde açılır. Şekil 4.17.'de merge_sort uygulaması için makefile dosyası gösterilmiştir.



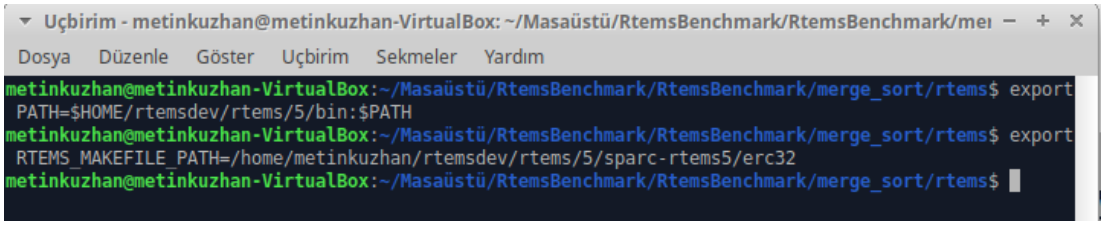
Şekil 4.17. RTEMS için merge.c' nin makefile dosyası

Programın RTEMS üzerinde çalıştırılması sonucu sıralması için dizi kodun içine gömülmesi gerekmektedir. Bunun için merge.c dosya içerisine veri girişi yapılır. Şekil 4.18.'de merge_sort uygulaması için test verilerinin girişi gösterilmişitir.



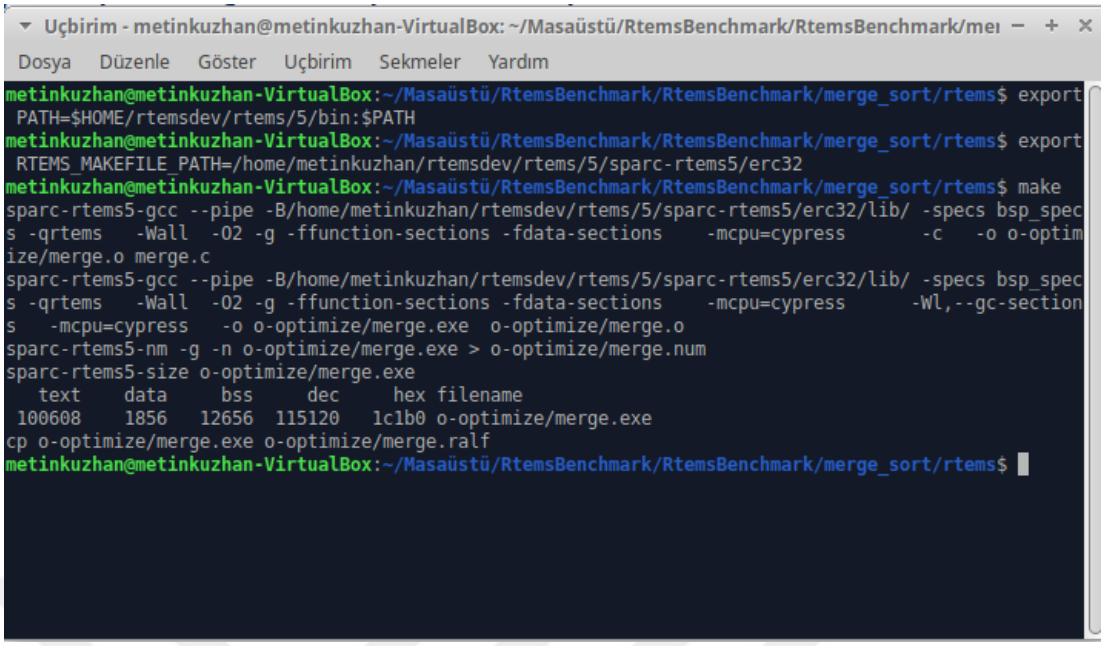
Şekil 4.18. RTEMS için merge. c' nin içine veri gömülmesi

RTEMS'nin çalışması için gerekli çevre değişkenleri ayarlanır. Derleme işleminin gerçekleşmesi için bu ayarlanmanın yapılması gerekir. Şekil 4.19.'da RTEMS üzerinde uygulamanın derlenmesi için gerekli çevre değişkenlerinin ayarlanması gösterilmiştir.



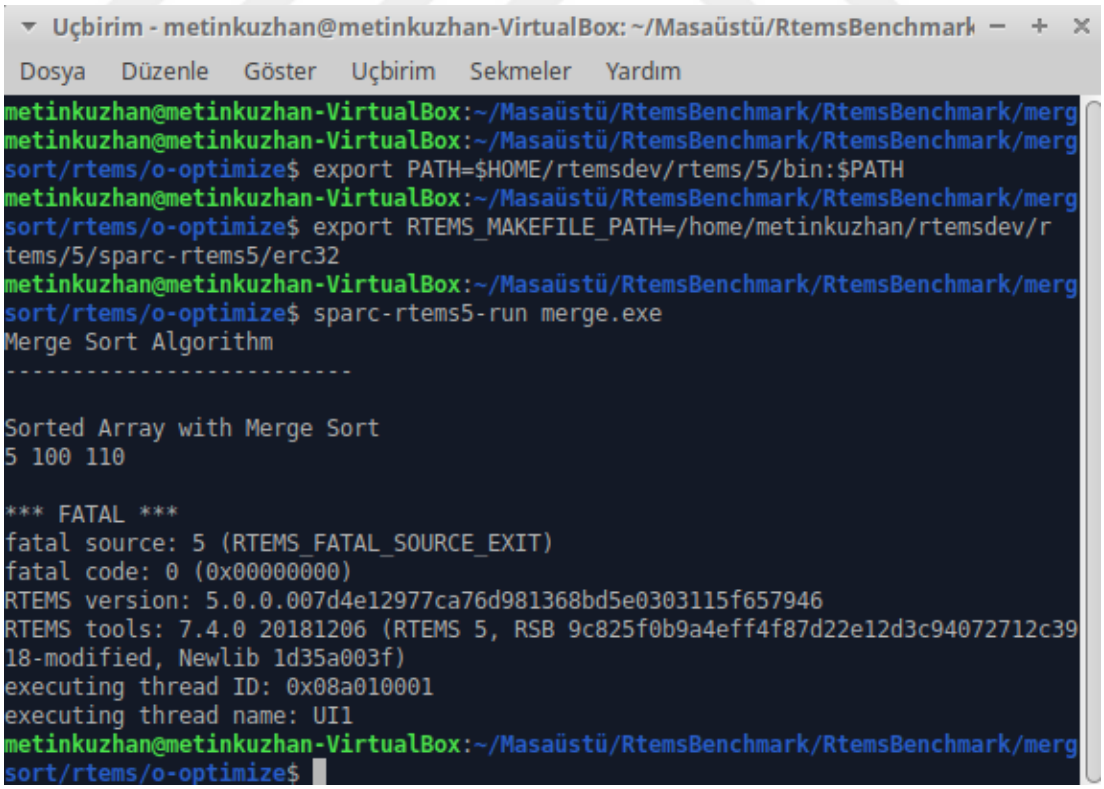
Şekil 4.19. RTEMS'nin çalışması için çevre değişkenleri ayarlanması

Terminal ekranı içerisinde make komutu verilerek merge.c dosyası derlenir. Şekil 4.20.'de merge_sort uygulamasının RTEMS gerçek zamanlı işletim sistemi üzerinde derleme işlemi gösterilmişitir.



Şekil 4.20. RTEMS'nin derleme işlemi gerçekleştirilmesi

Programın çalışması sonucunda sıralanmış dizi yazdıralacaktır. Şekil 4.21.'de merge_sort uygulamasının RTEMS üzerinden çalıştırılması sonucu oluşacak olan sıralanmış diziyi göstermektedir.



Şekil 4.21. RTEMS için merge uygulamasının çalışması oluşan sıralanmış dizi

BÖLÜM 5. TARTIŞMA

MBBench kıyaslama uygulamaları 2018 yılında toplandı. Yazılan uygulamalar Linux işletim sistemi üzerinde başarılı bir şekilde çalıştırıldı. Uygulamalar RTEMS üzerinde çalıştırılacak hale getirilirken dikkat edilen en önemli nokta kodların değişmemesi oldu.

MBBench kıyaslama takımı en kötü durum çalışma zamanı uygulamalarından oluşmaktadır. Uygulamaların kaynak kodları saf yapıdadır. Herhangi bir çalışma zamanında sonuç vermekten ziyade kodların çalışması üzerinde durulmuştur. Yazım aşamasında gerekli test işlemlerini gerçekleştirmek için çıktı işlemleri için kodlar yazılmıştır. Yayımlanmadan önce sonuç vericek olan kodlar yorum satırına alınarak MBBench githubdaki bulunan deposuna yüklenme işlemi yapılmıştır.

Bazı uygulamaların her iki platform üzerinde çalıştırılması sonucu Tablo 5.1.'de gösterilmiştir.

Çalışılan Süre(msn)					
İşletim Sistemi	bucket_sort	counting_sort	merge_sort	quick_sort	radix_sort
Xubuntu	0,37	1,14	0,34	7,7	0,3
RTEMS	0,25	0,28	0,29	0,35	0,15

Tablo 5.1. MBBench uygulamalardan bazılarının çalışma sonuçları

Tablo 5.1.'deki Xubuntu üzerindeki programların 10'ar kez çalışması sonucu elde edilen zamanlama sonuçlarının ortalaması alınmıştır. Aynı zamanda RTEMS üzerindeki elde edilen sonuçlar sparc simülatörü üzerinde 10'ar kez aynı test vektörünün programlar üzerinde çalışması sonucu elde edilmiş ortalama değerlerden oluşmaktadır.

Gerçek zamanlı uygulamamıza dışarıdan veri alırken 3 adet yöntem uygulandı:

- Komut satırı üzerinden girdi sağlama
- Rastgele veri üreterek girdi sağlama
- Herhangi bir dosya üzerinden girdi sağlama

Yukarıdı belirtilen 3 adet yöntem üzerinde Linux tarafında test işlemlerini yaparken hata ile karşılaşmadı. Fakat gerçek zamanlı uygulamaları sparc sanal işlemcisi üzerinde koşturulurken kıyaslama uygulamalarının yanlış sonuçlar ürettiği tespit edildi.

MBBench kıyaslama uygulamaların herbir uygulaması farklı bir girdi tiplerini desteklemektedir. Uygulamalar 3 adet girdi tipini içerir:

- Girdi Verisi
- Girdi Vektörü
- Girdi Dosyası

Uygulamaların dışarıdan değişken olarak veri alması için bu yöntemler kullanılmakta ısrar edilebilir. Bunu yapabilmek için Linux tarafında yazılan kodun çok değiştirilmesi gerekmektedir. Bu da çalışma konusunun dışında olan bir durumdur.

RTEMS üzerine değişiklik yapılan kodların test aşamasındaki yaşanılan sorunlar ayrıntılı olarak şu şekilde açıklandı:

5.1. Komut satırı üzerinde test vektörü alma

Linux tarafında yazılan uygulamaların RTEMS üzerinde uygulamak için algoritma kodları üzerinde gerekli değişikler yaptık. Yapılan değişikler sonucunda uygulamaları SPARC sanal işlemcisi üzerinde çalıştırmaya başlanmıştır. RTEMS ‘e göre yazılmış uygulamaların dışarıdan girdi verilerini alamadığını tespit edildi. İki

farklı işletim sistemi üzerinde çalışmanın gerçekleşmesinden kaynaklandığı için çalışmanın başarıya ulaşmadığı sonucuna varıldı.

5.2. Rastgele sayı üreterek test vektörü elde etme

Programlar girdi üretmek için tercih edilen diğer bir yöntem rastgele sayı üretmeyi kullanmaktır. Bunun için RTEMS uygulamaları üzerinde gerekli değişikler yapılarak programlar çalıştırılmaya başlandı. Yapılan çalıştırılma sonucunda rastgele üretilen girdilerin her çalışma da aynı girdilerin üretildiği görüldü. Bu oluşan durumun SPARC sanal işlemcisinden kaynaklı olduğu tespitine varıldı. Gerçek bir işlemci üzerinde sayı üretilmesi yapılmadığı için her seferinde aynı sayıların üretildiği görüldü.

5.3. Bir dosya üzerinde test vektörü elde etme

RTEMS uygulamaların test vektörü elde etmede tercih edilen bir diğer yöntemdir. C dili ile herhangi bir dosya formatında veri okumak kolaylıkla gerçekleştirebilen bir işlemdir. RTEMS örnek uygulamaları incelendiğinde bir dosyadan veri yazıp okuyarak işlemler gerçekleştirildiği tespit edildi. Fakat bu uygulama RTEMS üzerinde çalıştırıldığında etkilen veya oluşturulan gerçek bir dosyanın olmadığı görüldü. İşte bundan dolayı RTEMS de hazırlanmış olan uygulamalara bir dosya üzerinde veri elde edilmesi işlemi gerçekleştirilemedi. Algoritmaların RTEMS tarafından yazılan kodların çok değişmesi ile ancak veri alınabileceği tespit edildi.

Standard_deviation uygulaması girdi olarak dosya alabilen tek uygulamadır. Xubuntu üzerinde uygulama başarılı bir şekilde çalışmaktadır. Burada başarılı bir şekilde çalımasına rağmen RTEMS üzerinde harici girdi dosyası alarak uygulama çalıştırılamamıştır. RTEMS projesinin örnek uygulamaları incelendiğinde harici dosyadan veri alarak çalışan uygulama var olduğu görülmüştür. Fakat buradaki çalışan uygulamanın olmayan bir dosya üzerinde çalıştığı görüldü. Yani program kod içerisinde girilen değeri harici dosyaya yazmasına rağmen işletim sistemi üzerinde bir txt ya da herhangi bir format üzerinde çalışan dosya bulunamıştır. Bu durum asıl nedeni gerçek zamanlı sistemin bir dosya sistemi desteği olmamasından kaynaklıdır.

Sonuç olarak çalışmamızdan elde edilen önemli bulgular şu şekilde özetlenebilir:

- Test vektörü kullanan bir kıyaslama uygulama takımı MBBench elde edildi.
- Hem Linux hem de RTEMS tarafında kıyaslama yapabilen bir çalışma yapıldı.
- Harici dosyadan test vektörü elde etme konusunda RTEMS tarafında sorunlar yaşandı.
- Rastgele veri üreterek kıyaslama uygulamaları çalıştırılsada sanal Sparc işlemcisi RTEMS tarafından farklı rastgele üreterek test vektörü elde edilemedi.
- RTEMS tarafından kıyaslama uygulamalarına veri sağlamak için test vektörü komut satırından elde edilemedi.

BÖLÜM 6. SONUÇ

Bu tezde test vektörü alan kıyaslama uygulama takımı incelendi. Kıyaslama takımını C programlama dilini kullanarak yazıldı. Java programlama dili kullanarak yazılması tercih edilebilir.

Gerçek zamanlı uygulama yazma ortamı olarak RTEMS tercih edildi. Bunun muadili olan C dili kıyaslama uygulaması yazılabilecek olan ve destek veren Zephyr tercih edilebilir. Zephyr ile oluşturulacak olan kıyaslama takımı RTEMS ile yazılan arasında performans karşılaştırması yapılabilir.

Kıyaslama uygulama takımı yazarken komut satırında girdi olarak test vektörü almadan sorun yaşanmıştır. Bunun çözüm olarak Zephyr de komut satırı üzerinde girdi verisi alınarak kıyaslama takımı oluşturulabilir.

Kıyaslama takımı yazılırken rastgele veri üretilerek RTEMS üzerinde sabit test vektörü elde edildiği görüldü. Bunun çözümü için gerçek bir sistem üzerinde çalıştırılarak test vektörü üretimi sonuçlarına bakılabilir.

Kıyaslama uygulamalarının farklı kod yapılarını destekleyecek şekilde yazıldı. Bazı kıyaslama uygulamaları çok uzun kod satırları içerirken bazıları ise çok kısa kalmıştır.

MBBench 1.0 bütün uygulamalar tek çekirdekli sistemler de desteklenecek şekilde yazıldı. Günümüz sistemlerinin çok çekirdekli yapıları desteklediği düşünülürse kıyaslama uygulamalarının tamamının çok çekirdekli şekilde çalışacak hale getirilebilir.

Standart_deviation haricinde dışarıdan dosya alarak girdi alan daha başka algoritmalarda bulunarak kıyasama takımı zenginleştirilebilir.

MBBench 1.0 sadece C dili kullanılarak yazılan uygulamalardan oluştur. Bunun yanında RTEMS tarafıdan desteklenen Java ve Ada ile yazılması tercih edilebilir. Böylece C dili yazılmış alan uygulamalardaki sorunların ortaya çıkmasına bakılabilir.

KAYNAKLAR

[1] Laplante, P. A., Ovaska, S. J., Fundamentals of real-time systems. İçinde: Real-Time Systems Design and Analysis: Tools for the Practitioner, 4.Baskı, Wiley-IEEE Yayınevi, New Jersey, US, 1-26,2012.

[2] Kirner, R., Puschner, P., Discussion of misconceptions about wcet analysis. WCET Workshop, 2003.

[3] R. Project, RTEMS C User Documentation. Rtems.org, 2017.

[4] Buttazzo, G. C., Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications, 3. Baskı. Springer US Yayınevi, 1-519, 2011.

[5] Kopetz, H., Real-Time Systems: Design Principles for Distributed Embedded Applications. 2.Baskı. Springer US Yayınevi, 1-375, 2011.

[6] Fan, X., Introduction to Embedded and Real-Time Systems. İçinde: Real-Time Embedded Systems. 1.Baskı, Newnes Yayınevi, Oxford, 3-13, 2015.

[7] https://www.robomart.com/blog/real-time-operating-system-rtos., Erişim Tarihi:07.04.2019.

[8] Lamie, E. L., RTOS Concepts and Definitions. İçinde: Real-Time Embedded Multithreading Using ThreadX. 2.Baskı, Elsevier Yayınevi, Amsterdam,3-13, 2009.

[9] Peng, Z., Real-time operating systems. İçinde: Advanced Industrial Control Technology, 1.Baskı, Elsevier Yayınevi, Amsterdam, 613–683, 2010.

[10] Nandana, V., Jithendran, A., Shreelekshmi, R., Survey on RTOS: Evolution, Types and Current Research, International Journal of Computer Applications, 975–978, 2015.

[11] https://www.highintegritysystems.com/rtos/what-is-an-rtos., Erişim Tarihi:07.04.2019.

[12] https://www.rtems.org., Erişim Tarihi:13.01.2019.

[13] https://www.zephyrproject.org., Erişim Tarihi:23.01.2019.

[14] Massa, A. J., Embedded Software Development with eCos.1.Baskı. Prentice Hall Yayınevi, 1-399, 2003.

[15] https://www.ecos.sourceware.org/about.html., Erişim Tarihi:07.04.2019.

[16] A. Sood, Digging Inside the VxWorks OS and Firmware The Holistic Security. Secniche.Org, 2011.

[17] WIND RIVER, VxWORKS: The Safe and Secure RTOS for the Internet of Things.Windriver.com, 2016.

[18] Colin, A., Puaut, I., Worst-case timing analysis of the RTEMS real-time operating system. 13th Euromicro Conference on Real-Time Systems, Delft, Netherlands, 191-198, 2001.

[19] Islam, M. N., Extending WCET benchmark programs. Mälardalen University, School of Innovation, Design and Engineering, Yüksek Lisans Tezi, 2011.

[20] https://www.thegeekstuff.com/2012/02/rtos-basics., Erişim Tarihi:07.04.2019.

[21] Abella, J., Hernandez, C., Quiñones, E., Cazorla, F. J., Conmy, P. R., Azkarate-askasua, M., Perez, J., Mezzeti, E., Vardanega, T. WCET Analysis Methods: Pitfalls and Challenges on their Trustworthiness. 10th IEEE International Symposium on Industrial Embedded Systems, Siegen, Germany, 1-10, 2015.

[22] Gustafsson, J., Betts, A., Ermedahl, A., Lisper, B., The mälardalen WCET benchmarks: Past, present and future. 10th International Workshop on Worst-Case Execution Time Analysis, Brussels, Belgium, 136–146, 2010.

[23] https://www.absint.com/ait., Erişim Tarihi:23.01.2019.

[24] https://www.bound-t.com., Erişim Tarihi:13.04.2019.

[25] https://www.otawa.fr., Erişim Tarihi:13.04.2019.

[26] https://www.irisa.fr/aces/work/heptane-demo., Erişim Tarihi:13.04.2019.

[27] https://www.mrtc.mdh.se/projects/wcet., Erişim Tarihi:13.04.2019.

[28] Holsti, N., Saarinen, S., Status of the Bound-T WCET tool. Euromicro Worst-Case Execution Time Workshop 2002, Vienna, Austria, 36-41, 2002.

[29] Ballabriga, C., Cassé, H., Rochange, C., Sainrat, P., OTAWA: An open toolbox for adaptive WCET analysis. Lecture Notes in Computer Science, 2010.

[30] Li, X., Liang Y., Mitra T., Roychoudhury, A., Chronos: A timing analyzer for embedded software. Science Computer Programming, 2007.

[31] https://www.rapitasystems.com/products/rapitime., Erişim Tarihi:09.12.2018.

[32] https://www.rapitasystems.com., Erişim Tarihi:23.01.2019.

[33] Bernat, G., Colin, A., Petters, S., pWCET: A tool for probabilistic worst-case execution time analysis of real-time systems. Report-University York, 2003.

[34] https://www.rapitasystems.com/publications.html., Erişim Tarihi:26.12.2018.

[35] https://www.team.inria.fr/pacap/software/heptane., Erişim Tarihi:09.12.2018.

[36] Hardy, D., Rouxel, B., Puaut, I., The Heptane Static Worst-Case Execution Time Estimation Tool. 17th International Workshop on Worst-Case Execution Time Analysis, Dubrovnik, Crotia, 8-12, 2017.

[37] https://www.mrtc.mdh.se/projects/wcet., Erişim Tarihi:23.01.2019.

[38] https://www.netlib.org/benchmark/dhry-c., Erişim Tarihi:10.12.2018.

[39] https://www.spec.org/cpu2006., Erişim Tarihi:10.12.2018.

[40] Holsti, N., Långbacka, T., Saarinen, S., Using a Worst-Case Execution Time Tool for Real-Time Verification of the DEBIE Software. Proceedings of the Data Systems in Aerospace Conference, Montreal, Canada, 307-313, 2000.

[41] York, B. R., Benchmarking in context: Dhrystone. ARM white Paper, 1–7, 2002.

[42] Schoeberl, M., Preußer, T. B., Uhrig, S., The embedded Java benchmark suite JemBench. 8th International Workshop on Java Technologies for Real-time and Embedded Systems, Prague , Czech Republic, 120–127, 2010.

[43] https://www.eembc.org/about., Erişim Tarihi:05.12.2018.

[44] Poovey, J. A., Conte, T. M., Levy, M., Gal-On, S., A benchmark characterization of the EEMBC benchmark suite. IEEE Journals & Magazines, 18–29, 2009.

[45] Guthaus, M. R., Ringenberg, J. S., Ernst, D., Austin, T. M., Mudge, T., Brown, R. B., MiBench: A free, commercially representative embedded benchmark suite .2001 IEEE International Workshop on Workload Characterization, Austin, US, 3-14, 2001.

[46] https://www. vhosts.eecs.umich.edu/mibench., Erişim Tarihi:13.01.2019.

[47] Nemer, F., Cassé, H., Sainrat, P., Bahsoun, J., De Michiel, M., PapaBench: a Free Real-Time Benchmark. 6th International Workshop on Worst-Case Execution Time Analysis., Dresden, Germany, 1–6, 2006.

[48] Bienia, C., Kumar, S., Singh, J. P., Li, K. The PARSEC benchmark suite. Proceedings of the 17th international conference on Parallel architectures and compilation techniques, Toronto, Canada, 72-81,2008.

[49] Serttaş, S., Şahin, V. H., PBench: A Parallel, Real-Time Benchmark Suite. Acad. Perspect. Procedia, Antalya,178–186, 2018.

[50] https://www.mrtc.mdh.se/projects/wcet/benchmarks.html., Erişim Tarihi: 13.01.2019.

[51] Falk, H., Altmeyer, S., Hellinckx, P., Lisper, B., Puffitsch, W., Rochange, C., Schoeberl, M., Sørensen, R. B., Wägemann, P., Wegener, S., TACLeBench: A Benchmark Collection to Support Worst-Case Execution Time Research. 16th International Workshop on Worst-Case Execution Time Analysis, Toulouse, France, 1-6, 2016.

[52] https://www.github.com/tacle/tacle-bench., Erişim Tarihi: 16.04.2019.

[53] https://www.tacle.eu/index.php/activities/taclebench., Erişim Tarihi: 15.04.2019.

[54] https://www.xfce.org., Erişim Tarihi: 14.04.2019.

[55] https://www.xubuntu.org., Erişim Tarihi: 13.01.2019.

[56] Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C., Algoritmalara Giriş (Üçüncü Baskıdan Çeviri). 3.Baskı.Palme Yayıncılık, 2017.

[57] Mano, M. M. Bilgisayar Sistemleri Mimarisi (3. Basımdan Çeviri). 3.Baskı.Literatür Yayınları, 2015.

[58] https://www.cs.mcgill.ca/~rwest/wikispeedia/wpcd/wp/c/Caesar_cipher.htm., Erişim Tarihi: 14.04.2019.

[59] https://www.rtsrlab.sakarya.edu.tr/index_en.html., Erişim Tarihi:20.01.2019.

ÖZGEÇMİŞ

Metin KUZHAN, 02.10.1990'da Aksaray'da doğdu. İlk, orta ve lise eğitimini İstanbul'da tamamladı. 2008 yılında Abdurrahman ve Nermin Bilimli Anadolu Meslek Lisesi'nden mezun oldu. 2008 yılında başladığı Kocaeli Üniversitesi Teknik Eğitim Fakültesi Bilgisayar Öğretmenliği Bölümü'nü 2012 yılında bitirdi. Aynı yıl Van Muradiye Ünseli Mehmet Bey Çok Programlı Anadolu Lisesinde Bilişim Teknolojileri Öğretmeni olarak atandı. 2014 yılında Sakarya Üniversitesi Bilgisayar ve Bilişim Mühendisliği Bölümü'nde yüksek lisans eğitimine başladı. 2015 yılında Bahçelievler Türk Telekom Mesleki ve Teknik Anadolu Lisesinde Bilişim Teknolojileri öğretmeni olarak görev yapmaya başladı. 2018 yılında Kocaeli Üniversitesi Bilgisayar Mühendisliği Bölümünde Mühendislik tamamlama programını bitirdi. Halen İstanbul Bahçelievler Türk Telekom Mesleki ve Teknik Anadolu Lisesinde Bilişim Teknolojileri öğretmeni olarak görev yapmaktadır.