EĞİTSEL BİR BİLGİSAYAR MODELİ İÇİN YENİ BİR SİMÜLASYON SİSTEMİ

Namık Kemal KARASU

YÜKSEK LİSANS TEZİ
BİLGİSAYAR EĞİTİMİ

GAZİ ÜNİVERSİTESİ
BİLİŞİM ENSTİTÜSÜ

HAZİRAN 2012
ANKARA

Namık Kemal KARASU tarafından hazırlanan EĞİTSEL BİR BİLGİSAYAR MODELİ İÇİN YENİ BİR SİMÜLASYON SİSTEMİ adlı bu tezin Yüksek Lisans tezi olarak uygun olduğunu onaylarım.

Prof. Dr. Doğan ÇALIKOĞLU
Tez Yöneticisi

Bu çalışma, jürimiz tarafından oy birliği ile BİLGİSAYAR EĞİTİMİ Anabilim Dalında Yüksek lisans tezi olarak kabul edilmiştir.

Başkan : Yrd. Doç. Dr. Baha ŞEN

Üye : Prof. Dr. Doğan ÇALIKOĞLU

Üye : Yrd. Doç. Dr. Hüseyin ÇAKIR

Tarih : 19/06/2012

Prof.Dr. Halim SONCUL
Enstitü Müdürü

Bu tez, Gazi Üniversitesi Bilişim Enstitüsü tez yazım kurallarına uygundur.

TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, ayrıca tez yazım kurallarına uygun olarak hazırlanan bu çalışmada orijinal olmayan her türlü kaynağa eksiksiz atıf yapıldığını bildiririm.

Namık Kemal KARASU

EĞİTSEL BİR BİLGİSAYAR MODELİ İÇİN
YENİ BİR SİMÜLASYON SİSTEMİ
(Yüksek Lisans Tezi)

Namık Kemal KARASU

GAZİ ÜNİVERSİTESİ
BİLİŞİM ENSTİTÜSÜ
2012

ÖZET

Bu çalışmada, BB/3 (Bizim Bilgisayar 3) isimli eğitsel bir bilgisayar modeli için yeni bir yaklaşımla yeni simülasyon sistemi geliştirilmiştir. Simülasyon sisteminin temel özelliği alışılagelmiş izlence yazım ve hata ayıklama araçlarında olduğu gibi simgesel dilde yazılmış bir izlencenin yürütülerek donanım üzerindeki etkilerinin sergilenişinin yanı sıra daha alt düzeyde bilgisayar donanımını oluşturan sayısal devrelerce gerçekleştirilen işlemleri de simüle ederek ayrıntılı bir biçimde sergileyebilmesidir. Bu sayede yazılım ve donanım arasında ki ilişki bütün detayları ile ortaya konulabilmektedir. Yeni simülasyon sisteminde yazılımın donanım üzerindeki etkilerinin daha etkin bir biçimde izlenebilmesi için, simülasyon ekranında değişime uğrayan donanım birimleri vurgulanmakta, aynı zaman da önceki durumları da görüntülenebilmektedir. Simgesel dilde izlence yazabilmek ve hafıza muhtevasının izlenişi ve düzenlenişi için de yeni bileşenler geliştirilmiştir. Sadece belirli hafıza hücrelerinin izlenebilmesi için simülatöre gözetleyiciler eklenmiştir. Simülasyon sistemi tümüyle nesneye yönelimli programlama ilkelerine bağlı kalınarak geliştirilmiştir. Donanım birimlerini simüle eden nesneler arabirimden soyutlanarak tasarlanmıştır. Bu sayede esas simülatörü oluşturan yazılımın farklı kullanıcı arabirimlerinlerince de kullanımı mümkün kılınmıştır. Simülasyon sistemi Windows işletim sistemi için Delphi XE tümleşik geliştirme ortamında, Delphi dilinde geliştirilmiştir.

Bilim Kodu : **702.3.006**
Anahtar Kelimeler : **Donanım simülatörü, donanım tasarımı, kayıtlık aktarım dili, yığıt, birleştirici, nesne yönelimli tasarım**
Sayfa Adedi : **62**
Tez Yöneticisi : **Prof. Dr. Doğan ÇALIKOĞLU**

A NEW SIMULATION SYSTEM
FOR AN EDUCATIONAL COMPUTER MODEL
(M. Sc. Thesis)

Namık Kemal KARASU

GAZİ UNIVERSITY
INFORMATICS INSTITUTE
2012

ABSTRACT

In this study, a new simulation system has been developed for an educational computer which is called BB3 (Bizim Bilgisayar 3). The major features of the simulation system is just like the known program writing and debugging tools, by means of which can write a program in assembly language, run and watch hardware status changes and also can watch the operations made by digital circuits that forms whole hardware in details. Thus, a correspondence can be revealed between the hardware and software in all details. The new simulation system lets us see the effects of software over hardware more effectively by showing the changes on hardvare components with their previous status. The two new components had been developed for writing programs in assembly language and monitoring and editing the contents of memory. The Watches feature had been implemented to monitor the changes on certain memory cells. The whole simulation system has been developed under the object oriented programming paradigms. The objects that are responsible for simulating the hardware have been designed like an abstracted system. Thus, the major parts that consist the simulator can be used by different user interfaces and when a change is made on it, that can couse the interface to reflect that change. The simulation system had been developed in Delphi language with Delphi XE integrated development environment and it is compatible with Windows operation system.

Science Code : **702.3.006**
Key Words : **Hardware simulation, hardware design, register transfer language, stack, assembler, object oriented design**
Page Number : **62**
Adviser : **Prof. Dr. Doğan ÇALIKOĞLU**

TEŞEKKÜR

Çalışmalarım boyunca değerli yardım ve katkılarıyla beni sabırla yönlendiren hocam Sayın Prof. Dr. Doğan ÇALIKOĞLU'na, geliştirilen simülatörü test eden ve hataların giderilmesi hususunda bana yardımcı olan Gazi Üniversitesi Endüstriyel Sanatlar Eğitim Fakültesi'nin değerli öğrencilerine teşekkür ederim.

İÇİNDEKİLER

Sayfa

Sayfa

ÇİZELGELERİN LİSTESİ

Çizelge **Sayfa**

ŞEKİLLERİN LİSTESİ

Şekil **Sayfa**

RESİMLERİN LİSTESİ

Resim **Sayfa**

SİMGELER VE KISALTMALAR

Bu çalışmada kullanılmış bazı kısaltmalar, açıklamaları ile birlikte aşağıda sunulmuştur.

Kısaltmalar	**Açıklama**
BB/1	Bizim Bilgisayar 1
BB/2	Bizim Bilgisayar 2
BB/3	Bizim Bilgisayar 3
BBSD	Bizim Bilgisayar Simgesel Dili
Bİ	Birikeç
Ç	Çalıştır
ÇIKB	Çıkış Bayrağı
ÇBİ	Çıkış Bayrağı İzini
ÇIKK	Çıkış Kayıtlığı
D	Dolaylı Adres Biti
E	Elde
GBİ	Giriş Bayrağı izini
GİRB	Giriş Bayrağı
GİRK	Giriş Kayıtlığı
HAK	Hafıza Adres Kayıtlığı
HDK	Hafıza Destek Kayıtlığı
İŞ	İşlem Kodu
Kİ	Komut İşaretçisi
KK	Komut Kayıtlığı
KYV	Kesintiye Yol Ver
Yİ	Yığıt İşaretçisi

1. GİRİŞ

Bu çalışmada bilgisayar donanımı işleyiş ilkelerinin daha etkin bir şekilde öğretimi için geliştirilen bir yaklaşım olan Bizim Bilgisayar/3 (BB/3) isimli bilgisayar modeli [1] için yeni bir simülasyon sistemi hazırlanmıştır. Hazırlanan simülasyon sistemi ile; BB/3 için tanımlanmış olan Bizim Bilgisyar Simgesel Dili'nde (BBSD) [2] izlencelerin yazımı, bu izlencelerin birleştirilerek makine diline çevrimi ve sağlanan simülasyon ortamında izlencelerin yürütülerek donanım üzerindeki etkilerinin sergilenmesi mümkün kılınmıştır.

BB/3 donanım işleyişi ilkelerinin öğretiminde kullanılmak üzere, ÇALIKOĞLU (2002) tarafından daha önce BB/1 ve BB/2 isimleri altında tanıtılan [2], öğretim amaçlı bilgisayar modellerinin daha gelişkin bir şeklidir [1].

Bizim Bilgisayar, Model 1 (BB/1)

Bilgisayar donanımı ilkeleri gibi bir konunun genel boyutlarda ve sırf teorik olarak ele alınıp etkin bir tarzda öğretilmesi mümkün değildir. Bu nedenle bilgisayar donanım ilkeleri öğrencilere bir model üzerinde öğretilmelidir. Seçilecek model öğrencilerde tam bir güven hissi uyandırmak için gerçeğe uygun ve yeterince işlevsel olmakla birlikte gereğinden fazla karmaşık da olmamalıdır. Bu nedenle yeni modellerden birinin esas alınması pedagojik açıdan uygun değildir. Bu kriterler çerçevesinde Bizim Bilgisayar/1 modelinde, modern bilgisayarların temel işleyiş ilkelerini yeterince yansıtan PDP/8 bilgisayarı ölçü olarak alınmıştır [2].

1960'ların genel olarak en önemli mini bilgisayarı olarak kabul edilen PDP/8, Digital Equipment Corporation tarafından 22 mart 1965 yılında piyasaya sürülmüştür [3].

PDP/8 bilgisayarının ana hafızası her biri 12 bitten oluşan 4K kelimeden oluşmaktadır. Tüm aritmetik, mantık ve kaydırmaca işlemleri 12 bitlik "birikeç" isimli bir kayıtlıkta yapılır. BB/1'in PDP/8'e göre temel farkı, kelime uzunluğunun 12 bit yerine 16 bit

olmasıdır. BB/1 komutları PDP/8'in komutlarına benzer. BB/1'de hafıza adresleyen komutlar, kayıtlık komutları ve giriş çıkış komutları şeklinde üç tür komut vardır [2].

Kayıtlık devrelerini yazıyla tarif etmek için kayıtlık aktarım dili kullanılır. BB/1'in işleyişinin mantıksal tasarımında da kayıtlık aktarım dili kullanılmıştır.

Kayıtlıklara yüklü veriler üzerinde icra edilen işlemlere mikro işlem denir. Basit bir işlemdir ve bir saat zamanı içinde kayıtlıktaki veri üzerinde icra edilir. Mikro işlemin sonucunda kayıtlık içindeki ikili bilgi değişebilir veya başka bir kayıtlığa aktarılabilir [4].

Kayıtlıklar arasındaki veri aktarımını gerçekleştiren mikroişlemleri tanımlayan sembolik biçime, Kayıtlık Aktarım Dili adı verilir. Bunun anlamı istenen mikro işlemi yapacak donanımın mevcut olduğu ve işlemin sonucunun aynı veya farklı bir kayıtlığa aktarılabileceğidir. Kayıtlık aktarım dili, sayısal modüllerin kayıtlıkları arasında meydana gelecek mikroişlemleri sembolik biçimde ifade eden bir sistemdir. Sayısal bilgisayarların iç yapılarını hassas ve tam bir biçimde ifade etmek için uygundur [4].

Bu dildeki deyimler şu genel biçimde olmaktadır :

(Mantık İfadesi) : (mikroişlem), (mikroişlem), ... (mikroişlem)

Örneğin,

d_0z_1: HDK ← H, Kİ ← Kİ + 1

Bu deyimde, d_0z_1 şartı sağlandığı anda, HDK ← H ve Kİ ← Kİ + 1 mikro işlemlerinin gerçekleşeceği ifade edilmektedir [2].

Bizim Bilgisayar, Model 2 (BB/2)

Tasarlanan BB/1 bilgisayarı, biraz daha geliştirilmiş ve adına BB/2 denmiştir. BB/2'de iki adet yenilik yer almaktadır. İlk yenilik; kesinti düzenindeki kesinti talebi kaynaklarının ayrı ayrı izne bağlanmasıdır. İkinci yenilik ise; kayıtlık komutlarında kullanılan mikro işlemlerin zamanlarının yeniden düzenlenişidir. Bu düzenleniş ile birden fazla işlem aynı komutun içinde programlanabilmektedir [2].

Bizim Bilgisayar, Model 3 (BB/3)

BB/2' ye yığıt yapısı eklenerek BB/3 elde edilmiştir. Bu değişiklik sonucu BB/2' nin komut kümesine 7 yeni komut eklenmiş ve altyordam yapısı da değiştirilmiştir.

Yığıt bir hafıza parçasıdır ve son depolanan veri ilk geri dönen veri olur. Sayısal bilgisayarda yığıt temel olarak bir hafıza birimidir. Yığıta aktarılan son veri yığıtın en tepesinde yer alır. Yığıt bir adres kayıtlığı ile desteklenir. Bu tip adres kayıtlıklarına yığıt işaretçisi adı verilir. Çünkü bunun değeri daima yığıttaki son kelimenin adresini gösterir [4].

BB/3' te bu bağlamda hafızanın son kısmı yığıt olarak kullanılmaktadır ve mevcut kayıtlıklara Yığıt İşaretçisi (Yİ) adında 12 bitlik yeni bir kayıtlık eklenmiştir [1].

Bizim Bilgisayar Simgesel Dili

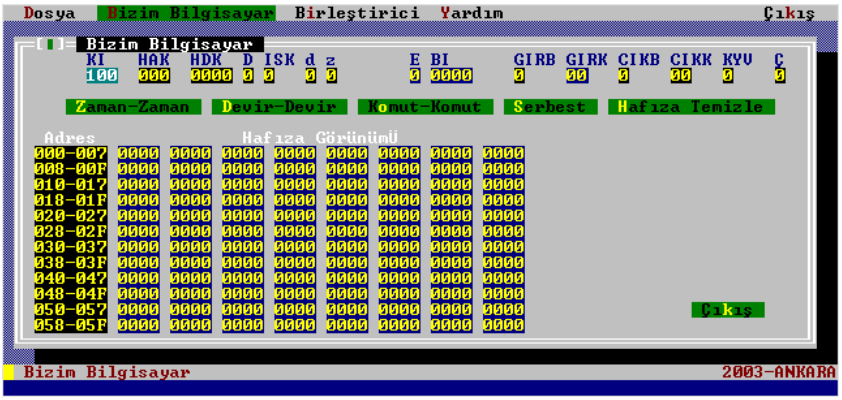
Bizim Bilgisayar Simgesel Dili (BBSD) ÇALIKOĞLU (2002) tarafından tanımlanmıştır [2]. BBSD dili basit fakat yeterince güçlü bir dildir. Bu dili kullanarak BB/3 komutlarıyla ciddi izlenceler yazılabilmektedir. Bu dil sayesinde öğrencilere "assembler" ilkelerinden söz etmek de mümkün olmaktadır.

BB/3 Simülasyon Sistemi

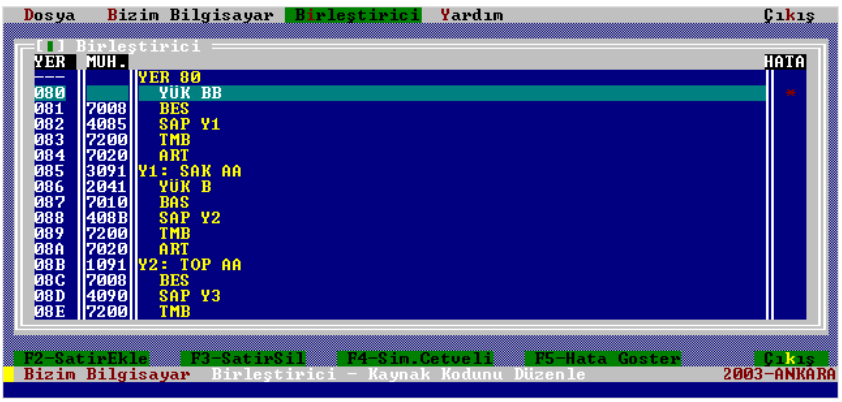
Simülasyon sistemi, BB/3'ün işleyişini sergilemek amacıyla geliştirilmiştir. Simülasyon sistem aracılığı ile BBSD' nde izlenceler yazılarak yürütülebilir ve izlencelerin mikro işlem düzeyinde BB/3 donanımı üzerindeki tesirleri incelenebilir.

Bu çalışmanın benzeri daha önce ÇAKIR tarafından 2003 yılında [5] ve ÇETİN tarafından 2009 yılında [6] yapılmıştır.

ÇAKIR tarafından gerçekleştirilen ilk çalışma Pascal dilinde Turbo Pascal ortamı kullanılarak MS-DOS işletim sistemi için geliştirilmiştir. Bu çalışma BB/2' yi destekleyen bir birleştirici ve donanım simülatöründen oluşur.

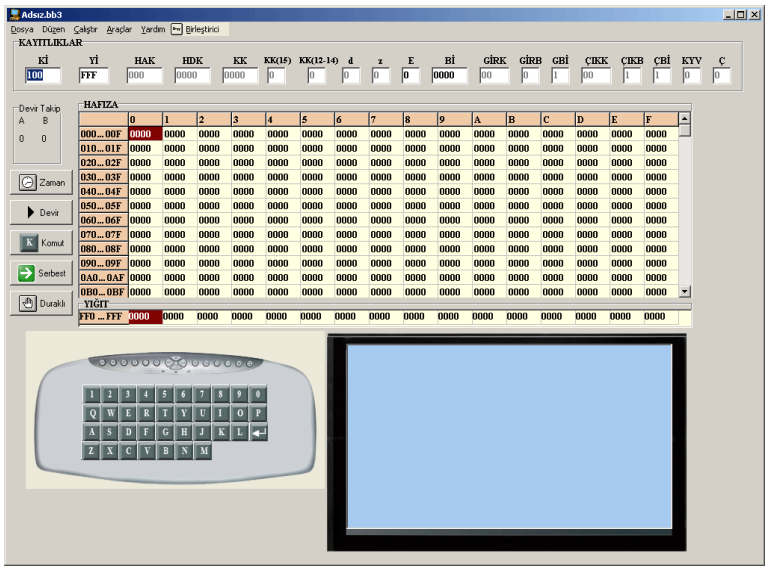


Resim 1.1. ÇAKIR tarafından geliştirilen simülatörün ekran görüntüsü

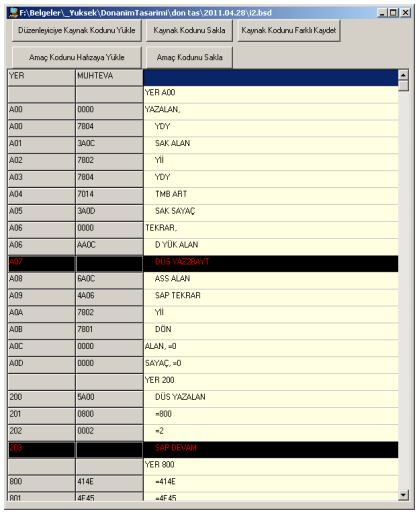


Resim 1.2. ÇAKIR tarafından geliştirilen birleştiricinin ekran görüntüsü

ÇETİN tarafından gerçekleştirilen ikinci çalışma ise Delphi dilinde, Delphi 7.0 tümleşik geliştirme ortamı kullanılarak MS Windows işletim sistemi için geliştirilmiştir. Bu çalışmada yine önceki gibi bir donanım simülatörü ve birleştiriciden oluşmaktadır. Farkı BB/3 donanımını simüle etmesidir. Bu amaçla simülatöre, yığıtı ve giriş - çıkış işlemlerini izleyebilmek için yeni bölümler eklenmiştir. Ayrıca ikilik, dörtlük, sekizlik, onluk ve onaltılık sayı sistemleri arasında çevrim yapılabilmesi için "Sayı Çevirici" bölümü eklenmiştir.

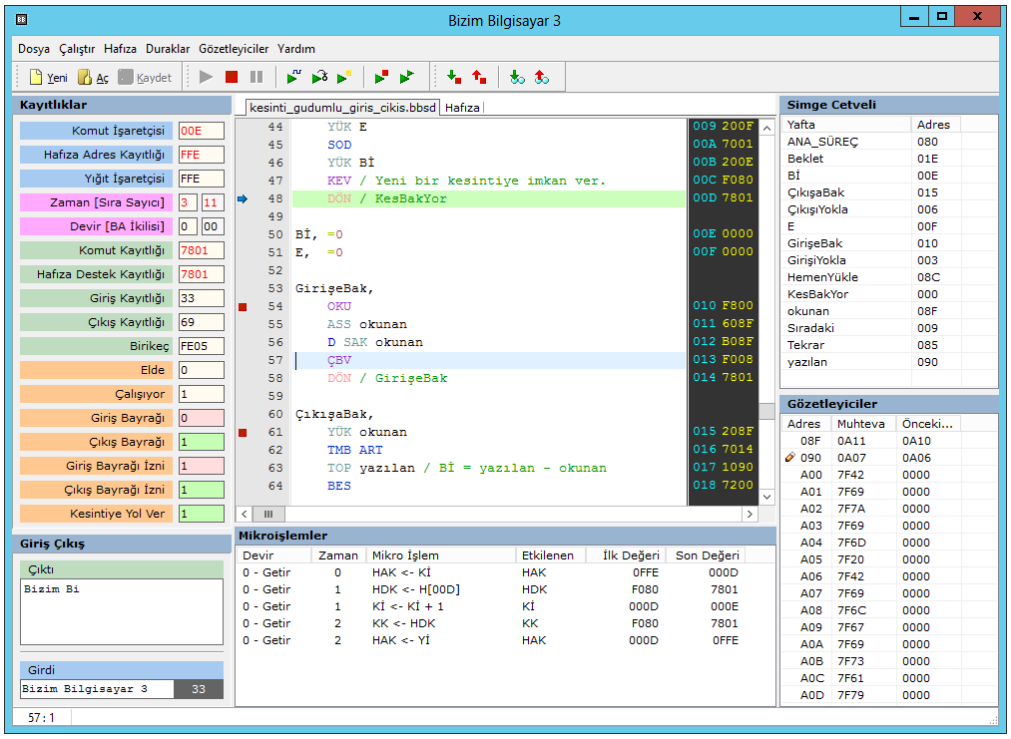


Resim 1.3. ÇETİN tarafından geliştirilen simülatörün ekran görüntüsü



Resim 1.4. ÇETİN tarafından geliştirilen simülatörün ekran görüntüsü

Yeni simülasyon sistemi ise yine Delphi dilinde fakat Delphi XE tümleşik geliştirme ortamı kullanılarak geliştirilmiştir. Yazılım önceki çalışmalarla tutarlılığı sağlamak amacıyla ve benzer hedefler çerçevesinde geliştirildiği için programlama dili olarak yine Delphi tercih edilmiştir.



Resim 1.5. Geliştirilen simülasyon sisteminin ekran görüntüsü

Yeni çalışmada yazılımın temel bileşenleri tek bir pencerede toplanmış, paneller ve sekmeler şeklinde düzenlenmiştir. Kayıtlıklar paneli işlemci kayıtlıklarının muhtevalarını ve muhtevalarında ki değişimleri göstermektedir. Simge cetveli paneli simgesel dilde yazılan izlencede ki yaftaları ve adreslerini göstermektedir. Giriş – çıkış paneli giriş ve çıkış işlemleri için kullanılabilmektedir. Gözetleyiciler paneli hafızanın belirli hücrelerinde ki değişimi izlemek için kullanılabilmektedir. Simgesel dil düzenleyici sekmesinde Bizim Bilgisayar Simgesel Dilinde izlenceler yazılabilir. Hafıza sekmesinde ise hafıza içeriği görüntülenebilir ve düzenlenebilir. İletiler panelinde izlence yazılırken yapılan yazım hataları görülebilir ve Mikro işlemler panelinde ise bilgisayar çalışıyorken bir komutun yürütülmesi süresince hangi mikro işlemlerin yapıldığı, bu işlemlerden hangi kayıtlıkların ne şekilde etkilendiği görülebilir.

Çalışmanın Başlıca Özellikleri ve Önemi

Yeni çalışmada mevcut sistem özellikleri esas alınarak onlarda olmayan özellikler belirlenmiş ve bu doğrultuda; bütün sistem baştan tasarlanarak mevcut sistemin bazı özellikleri farklı bir biçimde ortaya konmuş ve yeni özellikler eklenmiştir. Yazılım geliştirme yaklaşımı olarak nesne yönelimli programlama esas alınmış, sistem mümkün olduğunca modüler bir yapıda tasarlanmıştır. Arabirim ve simülasyon sistemi arasında ki ilişki nesnelerin olayları, yöntemleri ve özellikleri aracılığı ile sağlanmıştır. Böylesi bir yaklaşım sayesinde farklı bir arabirim geliştirilerek aynı simülasyon sitemi kullanabilir ya da tersi bir yaklaşımla simülasyon sisteminde ki değişiklikler arabirime müdahale edilmeksizin arabirime entegre edilebilir.

Simülasyon sisteminin bir diğer önemli özelliği de, izlence yazımı için yerleşik VCL bileşenleri arasında yer almayan iki yeni bileşenin bu sistemle birlikte geliştirilmiş olmasıdır. Bu bileşenlerden birincisi BBSD izlencelerinin yazımını sağlayan bir düzenleyicidir. İkincisi ise hafıza muhtevasını görüntülemek ve düzenlemek için kullanılan farklı bir düzenleyicidir.

Yeni simülasyon sisteminde ki değişiklikler ve yenilikler kısaca şu şekilde sıralanabilir;

1. Bütün sistem tek bir pencere üzerinden izlenebilir, yönetilebilir.
2. Birleştirici için, izlenceyi oluşturan dizgileri anlamlarına göre renklendirebilen, yazım hatalarını gösterebilen, simgesel dilde yazılan ifadelerin makine dili karşılıkları gösterebilen yeni bir bileşen geliştirilmiştir.
3. Hafıza muhtevasını izlemek ve düzenlemek için; muhtevayı farklı biçimlerde (ikilik, onluk, onaltılık sayı sistemlerinde ve simgesel dil karşılığı olarak) gösterebilen, hafızadaki belirli hücrelere hızlı bir biçimde geçişi sağlayan yeni bir bileşen geliştirilmiştir.
4. Hafızanın belirli hücrelerindeki değişimi izlemeye olanak sağlayan gözetleyiciler eklenmiştir. Gözetleyiciler hafıza muhtevasının hem son değerini ham de önceki değerini istenilen biçimde gösterebilmektedir.

5. Simge cetveli aracılığı ile simgesel dilde yazılmış izlencede ki yaftalar ve adresleri görülebilmektedir.
6. Kayıtlıklarda bir değişiklik meydana geldiğinde ilgili kayıtlık kırmızı renkle vurgulanmaktadır. Aynı zamanda muhtevası değişen kayıtlığın bir önceki muhtevası yine izlenebilmektedir.
7. İzlence yürütümü önceden olduğu gibi hem hafıza üzerinden hem de yeni bir özellik olarak simgesel dil düzenleyicisi üzerinden yapılabilmektedir.
8. Hafıza, önceki sistemlerde satır ve sütunlardan oluşan tablo benzeri bir yapıyla sergilenmekte iken yeni sistemde dikey tek bir sütun olarak tasarlanmış ve her bir hafıza hücresinin yanına bir de o hücredeki verinin farklı biçimlerde (ikilik sayı sisteminde, onluk sayı sisteminde işaretli tam sayı olarak, komut hatırası şeklinde) görüntülenebilmesini sağlayan yeni bir sütun eklenmiştir.
9. Bir diğer yeni özellik ise donanım çalışıyorken bir komutun icrası esnasında hangi mikro işlemlerin gerçekleştiğinin, o mikro işlemler sonucu hangi kayıtlıkların ne şekilde etkilendiğinin görülebilmesidir.
10. Yeni simülasyon sistemi bildik izlence yazım araçlarına bezerliğinden ötürü kullanımı kolay bir yazılımdır.

Bölüm 2’ de Bizim Bilgisyar/3 Bilgisayar Modeli, Bölüm 3’te Bizim Bilgisayar Simgesel Dili, Bölüm 4’te BB/3 Simülasyon Sistemi, Bölüm 5’ da Deneyler ve Simülasyon Sisteminin Kullanımı ve Bölüm 6’da da Sonuçlar, Öneriler ve İleride Yapılabilecek Çalışmalar anlatılmaktadır.

2. BİZİM BİLGİSAYAR/3 BİLGİSAYAR MODELİ

Bilgisayar donanımı ilkeleri gibi bir konunun genel boyutlarda ve sırf teorik olarak ele alınıp etkin bir tarzda öğretilmesi mümkün değildir. Bu nedenle bilgisayar donanım ilkeleri öğrencilere bir model üzerinde öğretilmelidir. Bilgisayar donanım tasarımı ve donanım işleyiş ilkelerinin daha etkili öğretilebilmesi için ÇALIKOĞLU (2002) tarafından bir bilgisayar tasarımı yapılmıştır [2].

Bu tasarlanan bilgisayar pedagojik açıdan mümkün mertebe basit, öğrencilerde tam bir güven duygusu oluşturmak ve gerçek izlenceler yazabilmek için gerçeğe uygun ve işlevsel olarak yeterince kabiliyetli olmalıdır. Bu nedenle yeni modellerden birinin esas alınması pedagojik açıdan uygun değildir. Bu kriterler çerçevesinde Bizim Bilgisayar/1 modelinde, modern bilgisayarların temel işleyiş ilkelerini yeterince yansıtan PDP/8 bilgisayarı ölçü olarak alınmıştır [2].

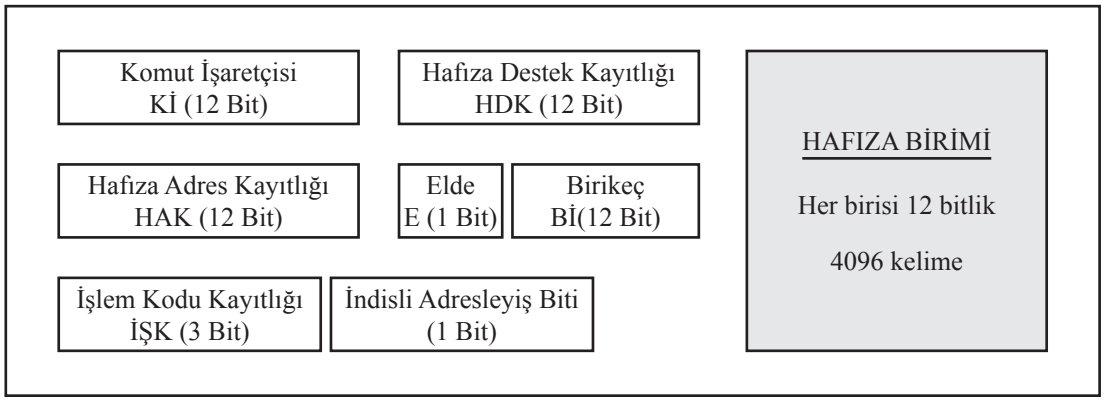
Bizim Bilgisayar/1 geliştirilerek sırasıyla "Bizim Bilgisayar/2" ve "Bizim Bilgisayar/3" modelleri tasarlanmıştır. Bu modellerin özellikleri sırasıyla bu bölümde ele alınacaktır.

2.1. PDP/8 Bilgisayarı ve Başlıca Özellikleri

PDP/8 bilgisayarı tasarımının nisbi sadeliğine rağmen simgesel boyutlarda ileri teknikleri andıran özelliklere sahiptir. Burada sunulan yaklaşımın geliştirilmesinde, benzer bir yöntem ortaya koyan Mano'dan da yararlanılmıştır (Mano 1982) [2].

PDP/8, 1960'lı senelerde Digital Equipment Corporation firması tarafından piyasaya sürülen ve dünyada mini-bilgisayar sınıfında önemli yer tutmuş olan bir bilgisayardır. Türkiye'ye ilk 1969 yılında girmiş ve ODTÜ Elektrik Mühendisliği Bölümü Bilgisayar Laboratuarına kurulmuştur. Ana hafızası her biri 12 bitlik olan 4K kelimeden oluşmaktadır. Bütün aritmetik, mantık ve kaydırmaca işlemleri, "Birikeç" denen 12 bitlik bir kayıtlıkta yapılmaktadır [2].

Resim 2.1. PDP/8 bilgisayarı [7]



Şekil 2.1. PDP/8 bilgisayarının ana hafızası ve temel kayıtlıkları

Şekil 2.1'de PDP/8 bilgisayarının temel kayıtlıkları ve ana hafızası yer almaktadır. Aritmetik işlem komutları, 12 bitlik işaretsiz tamsayılar içindir. Bu komutlar bir adet topla komutu, bir adet "2 tamlayanını al" komutu ve kaydırmaca komutlarından ibarettir. Programcılar işaretli tamsayıları temsil etmek için 2 – tamlayan aritmetiğini kullanmaktadırlar. Böylece işaret değiştiriş ve çıkartış mümkün olmaktadır. Çarpım, kaydırarak toplayışla, bölüm, kaydırarak çıkartışla elde edilmektedir. İki kelimelik ve ya daha uzun sayılarla işlemler, arada elde bitini kullanarak kelime

kelime gerçekleştirilmektedir. PDP/8, bir – adresli bir bilgisayardır ve komutları birer kelimeliktir. Toplam altı adet hafıza adresleyen komutu vardır. Bu komutlarda üç bit, işlem kodu olarak kullanılmaktadır. Geriye kalan dokuz bit ise, bir hedef adres belirlemekte kullanılmaktadır. Ancak bu dokuz bit, 4K'lık hafızayı doğrudan adreslemeye yetmediği için, indisli adresleyiş yöntemi ile "paging" yani sayfalandırış yöntemine müracaat edilmektedir. İndisli adresleyiş için özel bir indis kayıtlığı bulunmayıp, hafıza yerlerinin her biri bu amaçla kullanılabilmektedir [2].

2.2. Bizim Bilgisayar/1

Bir bilgisayarın işlevsel olarak yeterince kabiliyetli olabilmesi için makul bir çekirdek yapısına sahip olması ve komut kümesinin işlevsel olarak tam olması lazımdır. Bir komut kümesinin işlevsel olarak tam olması için aşağıda sayılan işlemlerin eksiksiz gerçekleştirilebilmesi gerekir [1].

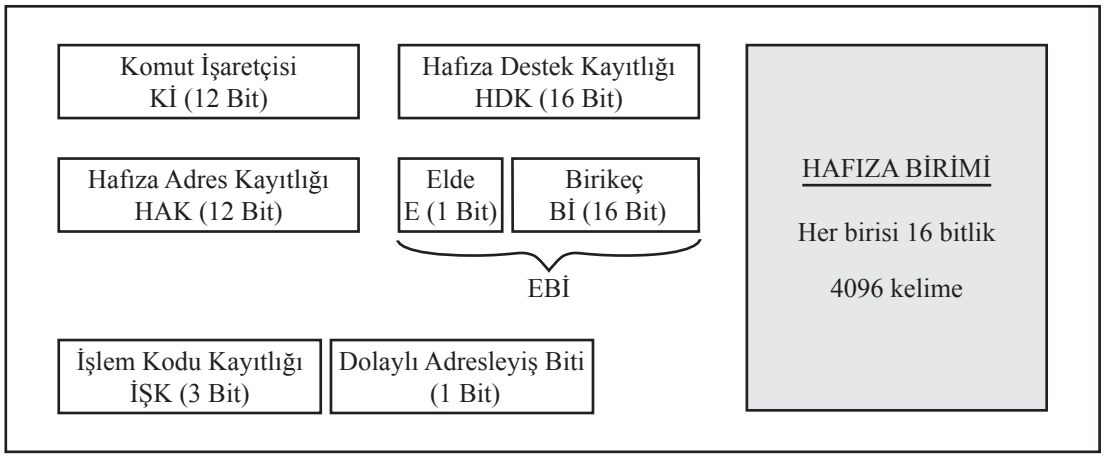
1. Aritmetik, mantık ve kaydırmaca işlemleri
2. Hafıza ile işlemci kayıtlıkları arasında veri alışverişi işlemleri
3. Komutların olağan işleniş sıralarını değiştiren işlemler (şartsız sapış)
4. Komutların olağan işleniş sıralarını işlem sonuçlarına ve durum bilgilerine göre değiştiren işlemler (şartlı sapış)
5. Giriş – Çıkış işlemleri

Bu işlemlerin yanı sıra, indisli adresleyiş yapılabilmesi ve altyordam çağrılabilmesi için kolaylıkların bulunması arzu edilen özelliklerdir.

Bizim Bilgisayar'ın ilk modeli BB/1'dir. PDP/8'e göre temel fark, kelime uzunluğunun 12 yerine 16 bit olmasıdır. BB/1 komutları, PDP/8'inkilere paraleldir. Her komut bir kelimedir ve bütün önemli PDP/8 komutlarının bir eşdeğeri vardır. Kelime uzunluğunun 16 bite çıkarılması sayesinde, işlem kodu ve indisli adresleyiş biti için dört bit ayrıldıktan sonra geriye kalan 12 bit ile 4K hafızanın tamamının doğrudan adreslenmesi mümkün olmaktadır [2].

2.2.1. BB/1'in temel kayıtlıkları

BB/1'in temel kayıtlıkları (Şekil 2.2) Kİ, HAK, İŞK, D, HDK, Bİ ve E'dir. Hafıza Adresi Kayıtlığı (HAK), hafızanın 4096 kelimesinden herhangi birine erişmek istenildiğinde, o kelimenin seçilmesi için, adresin yüklendiği kayıtlıktır. Hafıza Destek Kayıtlığı (HDK), hafızaya giriş–çıkış yapan verilerin geçmek zorunda oldukları bir ara kayıtlıktır. Birikeç (Bİ), bütün aritmetik, mantık ve kaydırmaca işlemlerini yapıldığı kayıtlıktır. Elde (E), Bİ kayıtlığından çıkan elde bitlerini tutmak için bir bitlik bir kayıtlıktır. Sıradaki komutu Komut İşaretçisi (Kİ) belirler. Kİ kayıtlığının gösterdiği komut hafızadan HDK'ya getirildiği zaman, arkadan gelen komutu göstermek için Kİ bir artırılır. İşlem Kodu Kayıtlığı (İŞK) ve D kayıtlıkları, hafızadan getirilen komutun ifası tamam olana kadar onun işlem kodunu ve dolaylı adres bitini muhafaza etmek içindir [1].



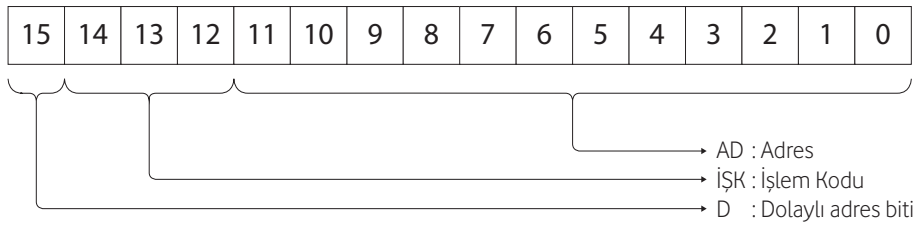
Şekil 2.2. BB/1 bilgisayarının ana hafızası ve temel kayıtlıkları

2.2.2. BB/1'in komut kümesi

BB/1"in komutları aşağıdaki gibi sınıflandırılmıştır [1]:

1. Hafıza adresleyen komutlar
2. Hafıza adreslemeyen komutlar
   a) Kayıtlık komutları
   b) Giriş – çıkış komutları

Hafıza adresleyen komut kelimelerinin düzeni Şekil 2.3.'de ki gibidir. İcap ettiğinde dolaylı adresleyiş yapabilmek için bir tane dolaylı adres biti (D), üç bitlik bir işlem kodu (İŞK) ve oniki bitlik bir adresten (AD) oluşur [1].



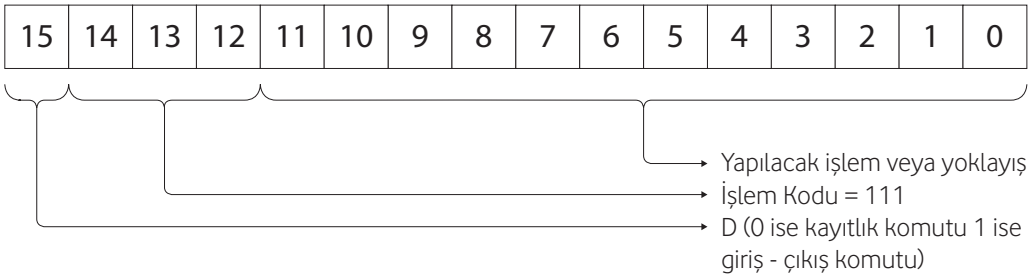
Şekil 2.3. Bir hafıza adresleyen komut kelimesinin genel şekli

Çizelge 2.1. BB/1'in hafıza adresleyen komutları

İşlem Kodu	Hatırası	Yaptığı İşlem	Açıklanışı
000=0	VE	Bİ ← Bİ ^ (h)	Hedef hafıza yerindeki kelimeyi birikeçle VE'le
001=1	TOP	EBİ ← Bİ ^ (h)	Hedef hafıza yerindeki kelimeyi (Elde E olarak) Bİ'ye TOP'la
010=2	YÜK	Bİ ← (h)	Hedef hafıza yerindeki kelimeyi Bİ'e YÜK'le
011=3	SAK	h ← Bİ	Bİ'yi hedef hafıza yerine SAK'la
100=4	SAP	Kİ ← h	Hedef hafıza adresine SAP
101=5	DÜS	h← Kİ, Kİ← h+1	Dönmek Üzere Sap
110=6	ASS	h← (h)+1, bununla (h)=0 olursa Kİ← Kİ+1	Artır, Sıfırsa Sek

Komutun esas hedefi olan adres “h” ile gösterilmektedir. D=0 hali, dolaysız (doğrudan) adresleyişi belirtir ve doğrudan, AD’ın kendisi h’dir. D=1 hali ise dolaylı adresleyişi belirtir ve bu durumda, AD ile gösterilen hafıza yerinin muhtevası olan kelime h’dir. Esas hedef olan h adresinde bulunan kelime de (h) ile gösterilmektedir. Hafıza adresleyen komutların işlem kodları, hatıraları (yani hatırlatan simgeleri), yaptıkları işlemler ve açıklanışları Çizelge 2.1’de görüldüğü gibi düzenlenmiştir [1].

Hafıza adreslemeyen komutların işlem kodları 111=7 dir. Hafıza adresleyen komutlar haricinde dolaylı adresleyiş söz konusu olamayacağından hafıza adreslemeyen komutlarda D biti, kayıtlık komutlarını giriş – çıkış komutlarından ayırmak için kullanılır. D bitinin “0” olması durumunda bu komut bir kayıtlık komutu olarak değerlendirilir, “1” olması durumunda ise komut giriş – çıkış komutu olarak değerlendirilir [1].



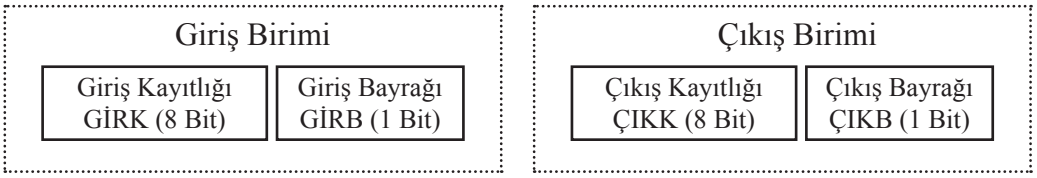
Şekil 2.4. Bir hafıza adreslemeyen komut kelimesinin genel şekli

Kayıtlık komutlarının en soldaki dört biti (0111) olarak sabitlenmiştir. Geriye kalan oniki bitin farklı birleşimlerine farklı anlamlar vererek 4096 adete kadar farklı komut tanımlamak mümkündür. Ancak büyük kod çözücülere ve aşırı karmaşıklığa gerek bırakmamak için bu oniki bitin her birine diğerlerinden bağımsız olarak Çizelge 2.2’de de görüldüğü şekilde işlemler tayin edilmiştir. Ayrıca Çizelge 2.2’nin her satırında; ilgili bitin hangisi olduğu, o bitin bir olmasına tekabül eden komut kodu, komut kodunun hatırası, komutun yaptığı işlem ve komutun açıklaması bulunmaktadır [1].

Çizelge 2.2. Kayıtlık komutları

İlgili Bit	Komut Kodu	Hatırası	Yaptığı İşlem	Açıklanışı
11	7800	SLB	Bİ ← 0	Bİ'yi sil
10	7400	SLE	E ← 0	E'yi sil
9	7200	TMB	Bİ ← Bİ'	Bİ'yi tamla
8	7100	TME	E ← E'	E'yi tamla
7	7080	SAD	Sağd EBİ	E ve Bİ'yi sağa döndür
6	7040	SOD	Sold EBİ	E ve Bİ'yi sola döndür
5	7020	ART	EBİ ← Bİ+1	Bİ'yi bir artır
4	7010	BAS	Bİ(15) = 0 ise Kİ ← Kİ + 1	Bİ artıysa sek
3	7008	BES	Bİ(15) = 1 ise Kİ ← Kİ + 1	Bİ eksiyse sek
2	7004	BSS	Bİ = 0 ise Kİ ← Kİ + 1	Bİ sıfırsa sek
1	7002	ESS	E = 0 ise Kİ ← Kİ + 1	E sıfırsa sek
0	7001	DUR		MİB'i durdur.

BB/1'de giriş birimi klavye, çıkış birimi de yazıcı olarak kabul edilmektedir. Bu birimlerle veri alış verişi yapmak için şekil 2.5'te görülen kayıtlıklar kullanılmaktadır [2].



Şekil 2.5. Giriş – çıkış işlemlerinde kullanılan kayıtlıklar

Giriş birimi olan klavyeden bir düğmeye basıldığında ilgili donanım o düğmeye tekabül eden Şekilcik kodunu GİRK'de hazır ettiği zaman kendiliğinden GİRB'yi "bir" yapar. GİRB'in "bir" olduğunu hisseden BB/1'in GBS isimli komutu vardır. Eğer GİRB=1 ise GİRK'de hazır olduğu anlaşılan kod bir komutla (OKU komutu) Bİ'ye alınır. Bu OKU komutu aynı zamanda GİRB'i "sıfır" yapar, böylece GİRK'deki okunmuş bilgi ile yerine gelecek yeni bilginin karıştırılması engellenir. Çıkış birimi olan yazıcıda yazılmakta olan bir şekilcik varsa, ilgili donanım onu yazmayı bitirip ÇIKK'ye yeni bir şekilcik kodu kabul etmeye hazır olduğu zaman kendiliğinden ÇIKB'yi "bir" yapar. BB/1'de ÇIKB'nin "bir" olduğunu hisseden bir komut (ÇBS) vardır. Eğer ÇIKB=1 ise, yazdırılacak bir şekilciğin kodu bir komutla (YAZ komutu) Bİ'den ÇIKK'ye verilir.

Bu YAZ komutu aynı zamanda ÇIKB'yi "sıfır" yapar ki, yazıcı donanımı yeni bir Şekilcik kodu verildiğini fark edip harekete geçsin, hem de yazıcının tekrar ne zaman ÇIKK'ye şekilcik kodu kabul etmeye hazır hale geleceği takip edilebilsin [1].

Bu esaslara göre oluşturulmuş olan giriş – çıkış komutları Çizelge 2.3'de listelenmektedir. Ayrıca Çizelge 2.3'ün her satırında; ilgili bitin hangisi olduğu, o bitin bir olmasına tekabül eden komut kodu, komut kodunun hatırası, komutun yaptığı işlem ve komutun açıklaması bulunmaktadır [1].

Çizelge 2.3. Giriş – çıkış komutları

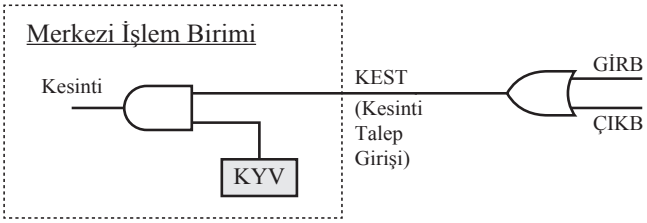
İlgili Bit	Komut Kodu	Hatırası	Yaptığı İşlem	Açıklanışı
11	F800	OKU	Bİ(7-0) ← GİRK, GİRB ← 0	Bİ'ye bir bayt oku
10	F400	YAZ	ÇIKK ← Bİ(7-0), ÇIKB ← 0	Bİ'den bir bayt yaz
9	F200	GBS	GİRB = 1 ise Kİ ← Kİ + 1	GİRB = 1 ise sek
8	F100	ÇBS	ÇIKB = 1 ise Kİ ← Kİ + 1	ÇIKB = 1 ise sek

Giriş – çıkış genelde iki türlü olur:

1. İzlence Güdümlü
2. Kesinti Güdümlü

Giriş – çıkış birimlerinin veri alışverişi için hazır olup olmadıklarının, sadece bayrak yoklayan komutlar vasıtasıyla ve izlence mantığı çerçevesinde takip edildiği giriş – çıkışa izlence güdümlü giriş – çıkış denir. Bu tür giriş – çıkışta diğer işlemler bırakılarak devamlı bayrak yoklandığı için ya da bayrak yoklamaları arasında diğer işlemler yapıldığı için ileri düzey uygulayışlarda bu yaklaşım yeterli olmaz. Birinci halde işlem zamanı israf edilmektedir, ikinci halde ise giriş – çıkış biriminin hazır olduğu hemen fark edilememekte, dolayısıyla da giriş – çıkış zamanı israf edilmektedir. Kesinti güdümlü giriş – çıkış'ta ise, ilgi gerektiren bir giriş – çıkış durumu ortaya çıktığı zaman, olağan izlencenin kesintiye uğratılması ve ilgi gerektiren durum tespit edilip gereği yapıldıktan sonra kalınan yere geri dönülebilmesi için tertibat alınır [1].

BB/1'de bir adet kesinti talep girişi vardır ve adına KEST denmektedir. KEST'e bayrak çıkışları şekil 2.6'da görüldüğü gibi bir "veya" geçidi ile bağlıdır. Kesinti izin denetimi için KYV (Kesintiye Yol Ver) isimli bir ikili vardır. Bir kesinti talebi, eğer ve ancak, KYV=1 ise kesinti doğurabilmektedir [1].



Şekil 2.6. Kesinti talep girişi ile ilgili donanım mantığı

Kesintiye bakan yordam, hafızanın en başında (yer 000'da) bir altyordam şeklindedir. Buraya sapış ve dönüş, aynı DÜS komutuyla olduğu gibi yapılmaktadır. Kesinti durumu doğduğunda son ifasına başlanmış olan komutun ifası bitince, olağan sıradaki komuta geçmeden önce (Sanki hemen orada varmış gibi) bir "DÜS 000" komutunun ifası gerçekleştirilir. Ayrıca kesintiye bakan altyordam çalışırken başka bir kesintiye izin verilememesi için kesintiye bakan yordama gidilmeden önce KYV← 0 yapılır. Kesintiye bakan altyordamdan dönmeden hemen önce KYV← 1 yapılır. Bu sayede yeni bir kesinti olabilmesi imkânı sağlanmış olur [1].

Gerektiğinde KYV'yi kurmak ve silmek için Giriş – Çıkış komutlarına iki komut ilave edilmiştir. Bu komutlar Çizelge 2.4'te listelenmiştir.

Çizelge 2.4. Kesintiye izin komutları

İlgili Bit	Komut Kodu	Hatırası	Yaptığı İşlem	Açıklanışı
7	F080	KEV	KYV ← 1	Kesintiye imkân var
6	F040	KEY	KYV ← 0	Kesintiye imkân yok

2.2.3. BB/1'de altyordam yapısı

Bir altyordama gitmek için "DÜS" komutu kullanılır. BB/1'de altyordamın ilk kelimesinde altyordamdan dönüş adresi tutulur. Altyordamın ikinci kelimesi altyordamın ilk komutudur. Alt yordamdan dönmek için ise "D SAP" komutu kullanılır. Aşağıda BB/1'de bir altyordam yapısı gösterilmektedir [1]:

```
/ "ALTYOR" isimli altyordam
YER 100
    ALTYOR:=0    / "DÜS" komutunun dönüş adresini saklaması için ayrılan
                 / yer.
    K1:          / "DÜS" komutunun hemen ardından ifa edilecek olan ilk
                 / altyordam komutu.

    / Altyordamın diğer komutları.

    D SAP ALTYOR / Altyordamdan geriye dönüş için kullanılan komut.

/ Altyordam çağrısı yapılan yer
YER 200
   ÇAĞRI: DÜS ALTYOR
```

2.2.4. BB/1'in işleyişi

BB/1 işleyiş tasarımında zaman-ölçülü bir yaklaşım tercih edilmiştir. BB/1'in işleyişinde zaman dayanağını bir sistem saat üreteci oluşturmaktadır. Her darbe bir zamanı, z_0, z_1, z_2, z_3 şeklindeki her dört zaman bir komut devrini oluşturmaktadır. Genel olarak her komutun ifası, birincisi "getir devri", ikincisi "ifa devri" olmak üzere iki komut devrinde gerçekleşmektedir. İndisli adresleyiş olduğunda, bu iki devir arasına bir "dolaylı devri" ilave edilmektedir. Kesinti meydana geldiğinde, o anda ifa edilmekte olan komutun "ifa devri"nden sonra, fazladan bir "kesinti devri" gerçekleştirilmektedir. Bu kesinti devri ile kesintiye bakan yordama geçilmesi sağlanmaktadır. Devirler, d0, d1, d2, d3 olarak belirtilmektedir [2]. Çizelge 2.5'te BB/1'in devir denetimi görülmektedir [1].

Çizelge 2.5. BB/1 bilgisayar modeli devir denetimi

Komut Devirleri	Kod Çözücü Çıkışı	Devir Kayıtlığı (B A)
GETİR devri (Komutu hafızadan)	d_0	0 0
DOLAYLI devri (Adresi çözüş)	d_1	0 1
İFA devri	d_2	1 0
KESİNTİ devri	d_3	1 1

Kayıtlık devrelerini yazıyla tarif etmek için kayıtlık aktarım dili kullanılır. BB/1'in işleyişinin mantıksal tasarımında da kayıtlık aktarım dili kullanılmıştır.

Kayıtlıklar arasındaki veri aktarımını gerçekleştiren mikroişlemleri tanımlayan sembolik biçime, Kayıtlık Aktarım Dili adı verilir. Bunun anlamı istenen mikro işlemi yapacak donanımın mevcut olduğu ve işlemin sonucunun aynı veya farklı bir kayıtlığa aktarılabileceğidir. Kayıtlık aktarım dili, sayısal modüllerin kayıtlıkları arasında meydana gelecek mikro işlemleri sembolik biçimde ifade eden bir sistemdir. Sayısal bilgisayarların iç yapılarını hassas ve tam bir biçimde ifade etmek için uygundur [4].

Bu dildeki deyimler şu genel biçimde olmaktadır :

(Mantık İfadesi) : (mikroişlem), (mikroişlem), ... (mikroişlem)

Örneğin,

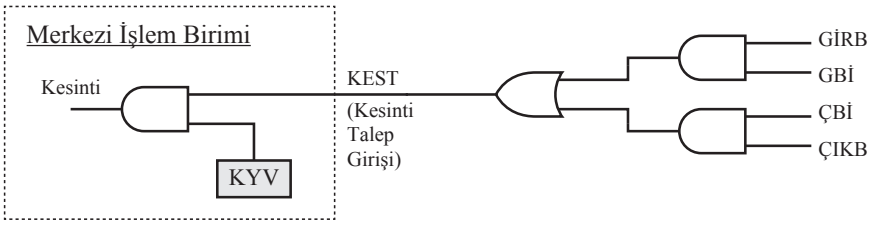
d_0z_1: HDK ← H, Kİ ← Kİ + 1

Bu deyimde, d_0z_1 şartı sağlandığı anda, HDK ← H ve Kİ ← Kİ + 1 mikro işlemlerinin gerçekleşeceği ifade edilmektedir. Yani, "getir devri"ndeki z_1 zamanında, bir yandan (HAK'ın gösterdiği adresteki) hafıza kelimesinin HDK'ya aktarılacağı, diğer yandan da, komut işaretçisinin bir artırılacağı bildirilmektedir.

2.3. Bizim Bilgisayar/2

Tasarlanan BB/1 bilgisayarı biraz daha geliştirilmiş ve adına BB/2 denmiştir. BB/2'de iki adet yenilik yer almaktadır:

İlk yenilik kesinti düzenindeki kesinti talebi kaynaklarının ayrı ayrı izine bağlanışıdır. Bu maksatla her bayrak için, GBİ (Giriş Bayrağı İzni) ve ÇBİ (Çıkış Bayrağı İzni) şeklinde birer "izin" ikilisi getirilmiştir ve KEST girişi şekil 2.7'de görüldüğü gibi KEST=GBİ•GİRB+ÇBİ•ÇIKB mantığıyla yeniden düzenlenmiştir. Bu yeni düzenlemeyle bayraklardan istenmeyenlerin kesinti talebinde bulunmaları önlenebilmektedir [2].



Şekil 2.7. BB/2'de kesinti talep girişi ile ilgili donanım mantığı

Kesinti talebindeki yeni düzenlemenin yapılabilmesi için BB/1 komutlarına Çizelge 2.6'da görüldüğü gibi 4 yeni komut eklenmiştir [1].

Çizelge 2.6. Bayrağa izin komutları

İlgili Bit	Komut Kodu	Hatırası	Yaptığı İşlem	Açıklanışı
5	F020	GBV	GBİ ← 1	Giriş bayrağı izni var
4	F010	GBY	GBİ ← 0	Giriş bayrağı izni yok
3	F008	ÇBV	ÇBİ ← 1	Çıkış bayrağı izni var
2	F004	ÇBY	ÇBİ ← 0	Çıkış bayrağı izni yok

BB/2'de ki ikinci yenilik ise kayıtlık komutlarında kullanılan mikroişlemlerin zamanları yeniden düzenlenmesidir. Böylece birden fazla işlem aynı komutun içinde

programlanabilmekte ve bu da komut kümesini daha da çok zenginleştirmektedir [2]. Çizelge 2.7'de kayıtlık komutlarının zamanlarının yenden düzenlenmiş hali görülmektedir [1].

Çizelge 2.7. BB/2'de Kayıtlık komutlarının işlem zamanları

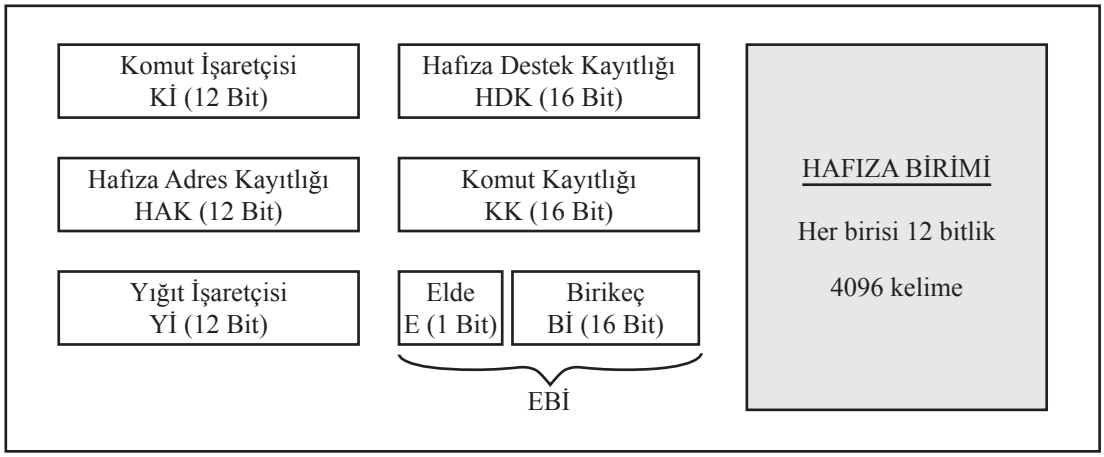
Komut Simgesi	İşleniş Zamanı	Komut Simgesi	İşleniş Zamanı
SLB	z_0	ART	z_2
SLE	z_0	BAS	z_0
TMB	z_1	BES	z_0
TME	z_1	BSS	z_0
SAD	z_3	ESS	z_0
SOD	z_3	DUR	z_3

2.4. Bizim Bilgisayar/3

BB/2 bilgisayarı ÇALIKOĞLU tarafından geliştirilerek yığıt yapısı ilave edilmiş ve altyordam çağrıları için yığıt kullanılan bir model elde edilmiştir. Bu yeni model BB/3 olarak adlandırılmaktadır [1].

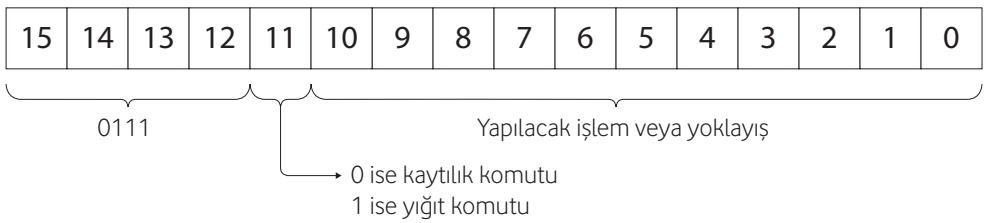
BB/3'te yığıt için hafızada belirli bir alan ayrılmıştır. Bu alan hafızanın en yukarısından (16 tabanındaki ifadesiyle, FFF_{16} adresli yeri) başlar ve yığıta veri konuldukça aşağı doğru büyür. Yığıtı izlemek için ise, BB/3 kayıtlıklarına Yığıt İşaretçisi (Yİ) adında yeni bir kayıtlık eklenmiştir. Bu kayıtlık yığıtın en tepesini işaret eder. Başlangıçta yığıt boş olacağı için Yİ kayıtlığı FFF_{16} adresini gösterir. Yığıta veri konulduğunda Yİ'de ki adres bir azaltılır. Yığıttan veri alındığında ise Yİ'de ki adres bir artırılır [1].

BB/3'e eklenen bir diğer kayıtlık ise o an yürütülen komutu içeren Komut Kayıtlığı'dır (KK). Komut kayıtlığının eklenmesi ile BB/2'de ki İŞK ve D kayıtlıklarına ihtiyaç ortadan kalkmıştır [1].



Şekil 2.8. BB/3 bilgisayarının ana hafızası ve temel kayıtlıkları

Yığıtın getirilmesiyle beraber BB/3'ün komut kümesine yedi yeni komut eklenmiştir. BB/1'de kayıtlık komutlarında en soldaki dört bit olan bitl5, bitl4, bitl3 ve bitl2 sabitlenmiş durumdadır (0111). Geriye kalan on iki bitin farklı birleşimlerine farklı anlamlar verilerek kayıtlık komutları oluşturulmuştur. BB/3'te ise eklemek istediğimiz 7 adet komutla daha fazla karmaşıklığa yer vermemek için, geriye kalan oniki bitin en solundaki bit11, yığıt komutlarını kayıtlık komutlarından ayırmak için kullanılmaktadır. Bir kayıtlık komutunun bit11 değerinin 1 olması, bu komutun bir yığıt komutu olduğunu göstermektedir [1].



Şekil 2.9. BB/3 bilgisayarının bir yığıt veya kayıtlık komut kelimesi

Yapılan bu değişiklik sonrasında kayıtlık komutları yeniden düzenlenmiştir. Kayıtlık komutlarının komut kodları ve zaman sıraları Çizelge 2.8'de görülebilir.

Çizelge 2.8. BB/3'te temel kayıtlık komutları ve işlem zamanları

İlgili Bit	Komut Kodu	Hatırası	Yaptığı İşlem	Açıklanışı	Zaman Sırası
10	7400	BAS	Bİ(15) = 0 ise Kİ ← Kİ + 1	Bİ artıysa sek	0
9	7200	BES	Bİ(15) = 1 ise Kİ ← Kİ + 1	Bİ eksiyse sek	0
8	7100	BSS	Bİ = 0 ise Kİ ← Kİ + 1	Bİ sıfırsa sek	0
7	7080	ESS	E = 0 ise Kİ ← Kİ + 1	E sıfırsa sek	0
6	7040	SLB	Bİ ← 0	Bİ'yi sil	1
5	7020	SLE	E ← 0	E'yi sil	1
4	7010	TMB	Bİ ← Bİ'	Bİ'yi tamla	2
3	7008	TME	E ← E'	E'yi tamla	2
2	7004	ART	EBİ ← Bİ+1	Bİ'yi bir artır	3
1	7002	SAD	Sağd EBİ	E ve Bİ'yi sağa döndür	4
0	7001	SOD	Sold EBİ	E ve Bİ'yi sola döndür	4

BB/3'e yeni eklenen yığıt komutlarından DÖN komutu altyordamdan geri dönüş için kullanılır. DÖN komutu geri dönüş adresini yığıttan alır. Bu adres Yİ'nin gösterdiği yerden alınarak komut işaretçisine yüklenir ve böylece geri dönüş sağlanır. Eğer yığıt boşken böylesi bir işlem yapılırsa Çalışıyor (Ç) durumu ikilisi 0 yapılarak işlemci durdurulur. DÖN komutunun bu özelliğinden dolayı BB/1'de ki DUR komutuna ihtiyaç ortadan kalkmıştır ve BB/3'ün komut kümesinden DUR komutu çıkartılmıştır.

Aşağıda BB/3'te bir altyordam yapısı gösterilmektedir [1]:

```
/ "ALTYOR" isimli altyordam
YER 100
    ALTYOR       / Altyordamın başlangıcı.
    K1:          / "DÜS" komutunun hemen ardından ifa edilecek olan ilk
                 / altyordam komutu.

    / Altyordamın diğer komutları.

    / Altyordama ait yerel veriler.

    DÖN          / Altyordamdan geriye dönüş için kullanılan komut.

/ Altyordam çağrısı yapılan yer
YER 200
   ÇAĞRI: DÜS ALTYOR
   DEVAM: ---    / Fiziksel sıradaki "DÜS" komutunu izleyen komut.
```

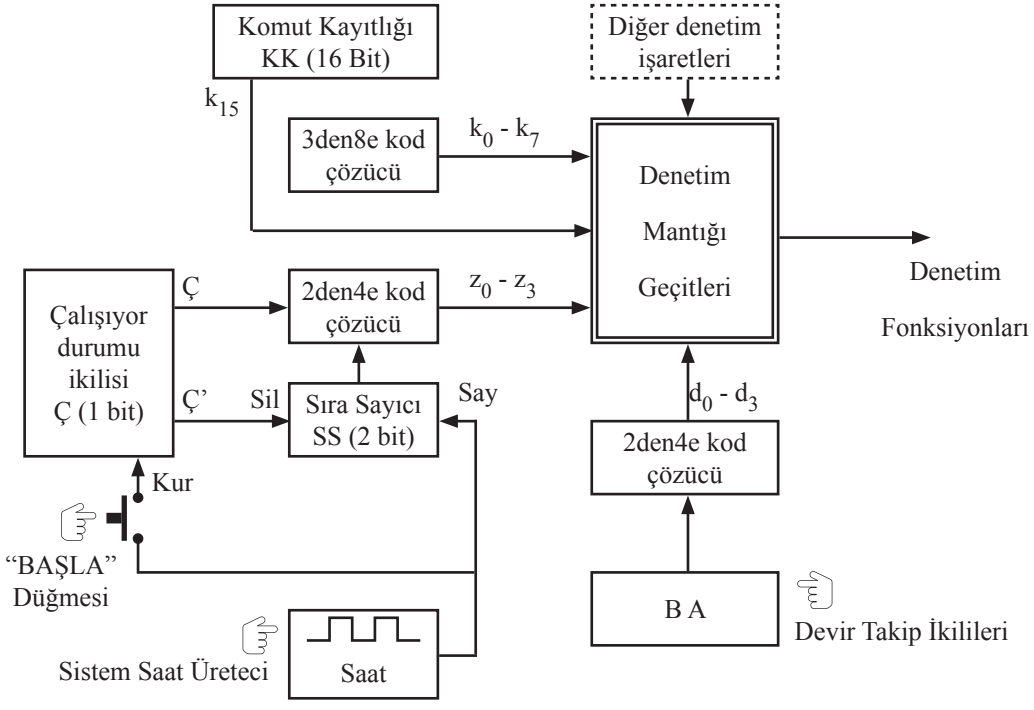
BB/3'e yeni eklenen yığıt komutları Çizelge 2.9'da görülebilir.

Çizelge 2.9. BB/3'te yığıt komutları

İlgili Bit	Komut Kodu	Hatırası	Yaptığı İşlem	Açıklanışı
6	7840	YİE	Yİ ← Yİ – 1	Yığıt işaretçisini eksilt
5	7820	YİA	Yİ ← Yİ + 1	Yığıt işaretçisini artır
4	7810	YVK	Yİ ← Yİ – 1, H(Yİ) ← Bİ	Yığıta veri koy(gönder)
3	7808	YVA	Bİ ← H(Yİ), Yİ ← Yİ + 1	Yığıttan veri al
2	7804	YDY	Bİ ← H(H(Yİ))	Yığıttan dolaylı yükle
1	7802	Yİİ	H(Yİ) ← H(Yİ) + 1	Yığıttaki İşaretçiyi İlerlet
0	7801	DÖN	Kİ ← H(Yİ), Yİ ← Yİ + 1	Altyordamdan dön.

Altyordam düzenindeki değişiklikden dolayı kesinti düzeni de değişmiştir. BB/3'te kesinti olduğunda, kesintiden sonraki dönüş adresi yığıta aktarılır [1].

BB/3 bilgisayar modeli denetim mantığı ilkesel görünümü şekil 2.9'da görülmektedir [1].



Şekil 2.10. BB/3 bilgisayar modeli denetim mantığı ilkesel görünümü

2.4.1. BB/3'ün işleyişinin mantıksal tasarımı

BB/3'te zamana ve devire göre yapılan mikroişlemler Çizelge 2.10'da gösterilmiştir [1].

Çizelge 2.10. BB/3'ün işleyişinin mantıksal tasarımı

KK(15)=v_{15}, KK(14)=v_{14}, ... , KK(0)=v_0 olsun.		
getir	d_0z_0	HAK ← Kİ
	d_0z_1	HDK ← H, Kİ ← Kİ + 1
	d_0z_2	KK ← HDK, HAK ← Yİ
	$(v_{15}k_7')d_0z_3$	A ← 1 {BA=01, d_1=1 olur.}
	$(v_{15}k_7')'d_0z_3$	B ← 1 {BA=10, d_2=1 olur.}
dolaylı	d_1z_0	HAK ← KK(AD)
	d_1z_2	HDK ← H
	d_1z_3	B ← 1, A ← 0
kesinti	d_3z_0	HDK(AD)←Kİ, Kİ←0, Yİ←Yİ-1
	d_3z_1	HAK←Yİ
	d_3z_2	H←HDK, KYV←0
	d_3z_3	B←0, A←0 {BA=00, d_0=1 olur.}

ifa

VE	$k_0d_2z_0$	HAK ← HDK(AD)	**TOP**	$k_1d_2z_0$	HAK ← HDK(AD)
	$k_0d_2z_1$	HDK ← H		$k_1d_2z_1$	HDK ← H
	$k_0d_2z_2$	Bİ ← Bİ ∧ HDK		$k_1d_2z_2$	EBİ ← Bİ + HDK
YÜK	$k_2d_2z_0$	HAK ← HDK(AD), Bİ ← 0	**SAK**	$k_3d_2z_0$	HAK ← HDK(AD)
	$k_2d_2z_1$	HDK ← H, Bİ ← (Bİ)'		$k_3d_2z_1$	HDK ← H
	$k_2d_2z_2$	Bİ ← Bİ ∧ HDK		$k_3d_2z_2$	Bİ ← Bİ ∧ HDK
SAP	$k_4d_2z_0$	Kİ ← HDK(AD)	**ASS**	$k_6d_2z_0$	HAK ← HDK(AD)
DÜS	$k_5d_2z_0$	HDK(AD)← Kİ, Kİ ← HDK(AD), Yİ ← Yİ - 1,		$k_6d_2z_1$	HDK ← H
				$k_6d_2z_2$	HDK ← HDK + 1
	$k_5d_2z_1$	HAK ← Yİ		$k_6d_2z_3$	H ← HDK ∧ HDK, Eğer (HDK=0) ise (Kİ ← Kİ + 1)
	$k_5d_2z_2$	H ← HDK			

p= $(v_{15})'k_7(v_{11})'d_2$, r_0= pz_0, r_1= pz_1, r_2= pz_2, r_3= pz_3 olsun.					
BAS	$v_{10}r_0$	Eğer (Bİ(15)=0) ise (Kİ ← Kİ + 1)	**SLE**	v_5r_0	E ← 0
BES	v_9r_0	Eğer (Bİ(15)=1) ise (Kİ ← Kİ + 1)	**TMB**	v_4r_1	Bİ ← (Bİ)'
BSS	v_8r_0	Eğer (Bİ=0) ise (Kİ ← Kİ + 1)	**TME**	v_3r_1	E ← E'
ESS	v_7r_0	Eğer (E=0) ise (Kİ ← Kİ + 1)	**ART**	v_2r_2	EBİ ← Bİ + 1
SLB	v_6r_0	Bİ ← 0	**SAD**	v_1r_3	Sağd EBİ
			SOD	v_0r_3	Sold EBİ

q= $(v_{15})'k_7(v_{11})d_2$, t_0= qz_0, t_1= qz_1, t_2= qz_2, t_3= qz_3 olsun.					
YİE	v_6t_0	Yİ ← Yİ - 1	**YVA**	v_3t_0	Bİ ← 0
				v_3t_1	HDK ← H, Bİ ← (Bİ)'
YİA	v_5t_2	Yİ ← Yİ + 1		v_3t_2	Bİ ← Bİ ∧ HDK, Yİ ← Yİ + 1
YVK	v_4t_0	Yİ ← Yİ - 1	**Yİİ**	v_1t_0	HDK ← H
	v_4t_1	HAK ← Yİ, HDK ← Bİ		v_1t_1	HDK ← HDK + 1
	v_4t_2	H ← HDK		v_1t_2	H ← HDK
YDY	v_2t_0	HDK ← H	**DÖN**	v_0t_0	HDK ← H
	v_2t_1	HAK ← HDK(AD), Bİ ← 0		v_0t_1	Kİ ← HDK(AD)
	v_2t_2	HDK ← H, Bİ ← (Bİ)'		v_0t_2	Yİ ← Yİ + 1
	v_2t_3	Bİ ← Bİ ∧ HDK		v_0t_3	Eğer (Yİ(11)=0) ise (Ç ← 0)

s= $(v_{15})k_7d_2z_3$ olsun.					
OKU	$v_{11}s$	Bİ(7-0) ← GİRK, GİRB ← 0	**KEY**	v_6s	KYV ← 0
YAZ	$v_{10}s$	ÇIKK ← Bİ(7-0), ÇIKB ← 0	**GBV**	v_5s	GBİ ← 1
GBS	v_9s	Eğer (GİRB=1) ise (Kİ ← Kİ + 1)	**GBY**	v_4s	GBİ ← 0
ÇBS	v_8s	Eğer (ÇIKB=1) ise (Kİ ← Kİ + 1)	**ÇBV**	v_5s	ÇBİ ← 1
KEV	v_7s	KYV ← 1	**ÇBY**	v_4s	ÇBİ ← 0

(Ortak Son) d_2z_3 : Eğer **(KYV∧KEST=1)** ise **(A←1)** {BA=11, d_3=1}, yoksa **(B←0)** {BA=00, d_0=1}

3. BİZİM BİLGİSAYAR SİMGESEL DİLİ

BB/1'in komutlarıyla ciddi izlencelerin yazılabilirliğini meydana çıkaracak örnekleri yazmakta ve onları okumakta kolaylık sağlamak için, kuralları belli bir simgesel dil gereklidir. Bu amaçla, ÇALIKOĞLU (2002) tarafından Bizim Bilgisayar Simgesel Dili (BBSD) isimli basit fakat yeterince güçlü bir simgesel dil tanımlanmıştır. Bu dil sayesinde öğrencilere "assembler" ilkelerinden söz etmek de mümkün olmaktadır [2].

BBSD; sabit simgeler, simgesel adres tanımları (Yaftalar), sabit değerler ve yardımcı komutlardan oluşmuştur [1].

Sabit Simgeler: Sabit simge cetvelinde yer alan 16 bitten oluşan değerler BBSD'de komut olarak adlandırılır. BBSD komutları Çizelge 2.11'de gösterilmektedir [1].

Çizelge 2.11. BBSD komutları

Simge	Değeri
ART	7004
ASS	6000
BAS	7400
BES	7200
BSS	7100
ÇBS	F100
ÇBV	F008
ÇBY	F004
D	8000

Simge	Değeri
DÖN	7801
DÜS	5000
ESS	7080
GBS	F200
GBV	F020
GBY	F010
KEV	F080
KEY	F040
OKU	F800

Simge	Değeri
SAD	7002
SAK	3000
SAP	4000
SLB	7040
SLE	7020
SOD	7001
TMB	7010
TME	7008
TOP	1000

Simge	Değeri
VE	0000
YAZ	F400
YDY	7804
YİA	7820
YİE	7840
Yİİ	7802
YÜK	2000
YVA	7808
YVK	7810

Simgesel Adres Tanımı (Yafta): Herhangi bir satırın solunda, bir harfle başlayan, virgülle veya iki nokta üst üste ile biten ve arada boşluk bulunmayan bir harf – rakam dizgisi yafta olarak tanımlanmıştır. Yaftanın değeri ise, bulunduğu satıra denk gelen adresin değerine eşittir [1].

Sabit değerler: BBSD de sabit değerler işaretli veya işaretsiz 16'lık sayı olarak yazılabilir. Sabit değer eksi işaretli olduğunda, esas muhteva sabit değerin 2-tamlayanı olur [1].

Yardımcı Komutlar: BBSD'de "YER", "SON" ve "/" yardımcı komutları kullanılır. YER yardımcı komutu, makine komutlarının hafızada denk geleceği yeri belirler. Son yardımcı komutu, simgesel izlencenin sonunu belirtir. "/" yardımcı komutu, izlenceye açıklayıcı ifadeler eklemeyi sağlar [1].

3.1. BBSD'nin tanımı

BBSD dilini tanımlarken ölçü, sadece çok lüzumlu özelliklerin bir araya getirilmesidir [1]. Aşağıda bu dile ait özellikler sıralanmıştır [2].

1. BBSD dili tanımı boyunca, "kelime" dendiğinde bir hafıza yeri muhtevası (komut veya veri) kastedilmektedir [1].
2. BBSD'de sayılar 16 tabanında ifade edilirler [2].
3. Her satırda en çok bir hafıza yeri muhtevasının (komut veya veri) ifadesi olabilir [2].
4. Makine komutlarının hafızada denk geleceği yerin tayini için, sıradaki makine komutlarının hangi yerden itibaren dizilmesi istendiği, bir YER yardımcı komutuyla belirtilir [2]. Ör. YER 200
5. Makine komutlarının sayısal kodları yerine simgeleri yazılabilir [2].
6. Bir hafıza yerinin muhtevası bir simge olarak, veya işaretli/işaretsiz bir 16'lık sayı olarak yazılabilir. Bu sayı eksi işaretli olduğunda, esas muhteva, o sayının 2-tamlayanı olur. İşaretsiz 16'lık sayıları simgelerden ayırt edebilmek için önlerine bir "=" işareti konur. Örneğin: BABA, bir simgedir, =BABA ise 16'lık bir sayıdır [2].
7. Bir satıra birden fazla komut, 16'lık sayı veya simge, aralarında boşluk bırakılmak suretiyle yazıldığında o satıra tekabül eden muhteva, bunların 16 bitlik karşılıklarının VEYA'lanmasıyla elde edilen sonuç olur [2].
8. Her hangi bir satırın solunda, bir harfle başlayan, virgülle veya iki-nokta-üst-üste ile biten ve arada boşluk bulunmayan bir harf-rakam dizgisi, o satıra denk gelen adres değerine sahip bir yafta (simgesel adres tanımı) olur. Böyle bir yafta ile tanımlanan bir simgesel adres, hafıza adresleyen komutlarda kullanılabilir [2]. Bir satırda yalnız yafta tanımlanabileceği gibi, yafta tanımlandıktan sonra, başka simgesel

ifadeler de gelebilir. Yalnızca yafta tanımlanan satırlarda hafızaya herhangi bir değer aktarılmasına gerek kalmadığı için yer değeri artırılmaz. Herhangi bir satırda tanımlanan yafta simgesi başka bir satırda tekrar tanımlanamaz [6].

Örnek:

```
K1: ART
---
SAP K1
```

9. Simgesel programa açıklayıcı ifadeler ilave edebilmek için "/" işareti kullanılır. Bu işaretin sağında kalan şekilcikler makine diline çevrilirken dikkate alınmaz.
10. Hafıza adresleyen komutlarda hatıradan önce bulunan bir D harfi, indisli adresleyişin olduğunu belirtir [2].
11. Simgesel programın sonunu belirtmek için SON yardımcı komutu kullanılır.

BBSD dilinde yazılmış bir izlence örneği [2]:

```
/ Bu izlence, iki (işaretli) tamsayının büyük olanını birikece koyduktan
/ sonra durur.
YER 200
    YÜK B
    TMB     / B'nin 2-tamlayanını
    ART     / oluştur.
    TOP A   / Bİ = A-B
    BES
    SAP Y1  / A büyük (veya B'ye eşit)
    YÜK B   / B büyük
    SAP Y2
Y1: YÜK A
Y2: DÖN

YER 2F0
A:  =A2
B:  =8C
    SON
```

4. BİZİM BİLGİSAYAR SİMÜLASYON SİSTEMİ

BB/3 simülasyon sistemi Delphi XE ortamında nesneye yönelik programlama ilkeleri doğrultusunda geliştirilmiştir. Kod yazım esasları olarak Delphi geliştiricileri tarafından belirlenen esaslar dikkate alınmıştır. Bu esaslardan bazıları şu şekildedir [8].

Adların yazımı (Naming conventions): Tamamı küçük harflerle yazılan ayrılmış kelimeler ve yönergeler hariç bütün Pascal tanımlayıcıları, her bir kelimesinin ilk harfi büyük diğer harfleri küçük olacak şekilde yazılmalıdır.

Sınıfların ve Türlerin tanımlanışları: Bir sınıf tanımı öncesinde iki boşluk ve ardından "T" harfi ile başlar. Sınıf adındaki her bir kelimenin ilk harfi büyük harfle yazılır. Tür tanımları içinde aynı şey geçerlidir.

Örnek:
```
type
  THafiza = class
  //...
```

Alan isimlerinin yazımı: Bir sınıf içindeki bütün alan isimleri "F" ön eki ile başlar. Alan adındaki her bir kelimenin ilk harfi büyük harfle yazılır. Alan tanımları sınıf tanımının "private" bölümünde yapılır.

Örnek:
```
type
  THafiza = class
  private
    FMuhteva: array [0..4095] of Word;  //...
```

Sıralı türlerin (Enumerated types) yazımı: Sıralı türe ait her bir öğe, ait olduğu tür adındaki kelimelerin ilk harflerinin küçük harf şeklindeki yazımı ile başlar. Devamında ise genel kurala uyulur.

Örnek:
```
type
  TBBSDParcaTuru = (ptBosluk, ptHic, ptYorum, ptYafta, //...
  // ...
```

Yöntem adlarının yazımı: Yöntem adlarının yazımında da genel kural geçerlidir. Yöntem adındaki kelimeleri birbirinden ayırmak için alt çizgi kullanılmamalıdır. Ayrıca yöntem adları bir yüklem içermelidir.

Örnek:

```
type
  TIslemci = class
  // ...
  public
    procedure Sifirla;
    procedure ZamanZamanCalistir;  // ...
```

Bazı durumlarda bu kuralların dışına çıkılmıştır. Örneğin bir özelliğin değerini belirleyen yordam kurallar gereği "Set" ön eki ile başlar, değerini geri döndüren işlev ise "Get" ön eki başlar. Olay yordamlarının adları ise "On" ön eki ile başlar. Bu gibi yazımlarda "Set" yerine "yaz", "Get" yerine "oku" ön ekleri kullanılmıştır. Olay yordamlarının adları ise "On" ön eki ile başlamak yerine "Ol" son eki ile sonlandırılmıştır.

Örnek:

```
type
  TBBSDIzlence = class(TStringList)
  // ...
  public
    property YaftaEkleninceOl: TYaftaEklenince
      read FYaftaEkleninceOl write FYaftaEkleninceOl;
    property Degisti: Boolean read FDegisti write yazDegisti;
  // ...
```

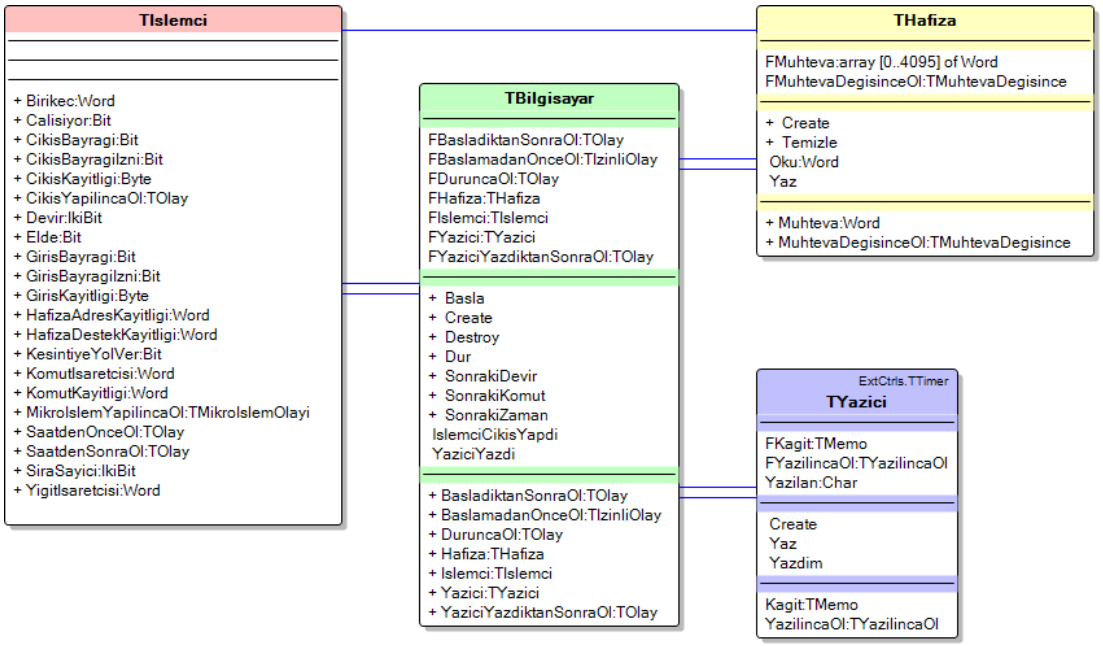
BB/3 Simülasyon Sistemi daha önceki çalışmalardan farklı olarak modüler bir yapıda yeniden tasarlanmış ve kodlanmıştır. Üçüncü parti Delphi bileşenlerinin kullanımından özellikle kaçınılmıştır. Bu sayede bileşen programcılığı gibi ileri düzey tekniklerde deneyim kazanılmış, üçüncü parti bileşen kullanımının doğuracağı (telif hakları, sürekli erişilebilirlik gibi...) riskler ortadan kaldırılmış ve özgün bir çalışma ortaya konmuştur.

Yeni sistem altı modülden oluşmaktadır. Her bir modül mümkün olduğunca birbirinden yalıtılmıştır. Örneğin kullanıcı arabirim sınıfları simülatör sınıflarının özelliklerine, yöntemlerine veya olaylarına erişebilirken tam tersi mümkün değildir. Bu sayede simülatör sınıfları farklı ortamlarda da kullanılabilir kılınmıştır.

Yeni sistemi oluşturan modüller aşağıdaki gibidir.

4.1. Simülator Modülü

Donanım simülasyonunu gerçekleştiren sınıfları içerir. Bu sınıflar TIslemci, THafiza, TYazıcı ve TBilgisayar sınıflarıdır. Simülatör sınıfları arasındaki ilişki aşağıdaki diyagramda görüldüğü gibidir.



Şekil 4.1. Simülatör sınıfları ilişki diyagramı[1]

TBilgisayar sınıfı simülatör modülünün dışarıya açılan penceresidir. Kullanıcı arabirim sınıfları simülatör modülü ile bu sınıf üzerinden iletişim kurar.

TIslemci sınıfı BB/3 bilgisayarının işlemcisini simüle eder. Kayıtlıklar, mikroişlemler, denetim fonksiyonları vs. bu sınıfta simüle edilmektedir.

THafiza sınıfı BB/3 bilgisayarının hafızasını simüle eder. Bu sınıfa hafızadaki değişimleri izleyebilmek için bir de olay eklenmiştir.

1 TIslemci sınıfına ait yöntemler ve alanlar diyagramda gösterilmemektedir.

Olay (event – driven) güdümlü programlamanın ardındaki temel fikir, belirli olayların uygulamanın akışını denetlemeleridir. Bir program zamanının büyük kısmını bu olayları bekleyerek geçirir ve bunların pek çoğuna tepki olarak kod sunar. Örneğin kullanıcı fare düğmelerinden birine tıkladığında bir olay meydana gelir. Bu olayı açıklayan bir mesaj, halen imlecin altında olan pencereye gönderilir. Bu pencere için olaylara tepki veren kod da, bu olayı alarak onu işler ve uygun bir biçimde tepki verir. Program bu olaya tepki vermesini bitirdiğinde, bekleme veya atıl durumuna geri döner [9].

THafiza sınıfına eklenen "MuhtevaDegisinceOl" olayı kullanıcı arabiriminin hafızadaki değişiklikleri izlemesine yardımcı olur. Bu olay aşağıdaki gibi tanımlanmıştır.

```
type
  TMuhtevaDegisince = procedure (const Adres, YeniMuhteva,
    EskiMuhteva: Word) of object;
```

Yukarıdaki tanım, bir tür tanımıdır. Asıl olay işleyici yöntemi belirlemek için ilgili sınıf içinde onu işaret eden bir işaretçi tanımlanmalıdır. "TMuhtevaDegisince" türündeki bu işaretçi Thafıza sınıfında aşağıdaki gibi tanımlanmıştır.

```
// ...
  THafiza = class
  private
    FMuhtevaDegisinceOl: TMuhtevaDegisince;
    // ...
  public
    property MuhtevaDegisinceOl: TMuhtevaDegisince
      read MuhtevaDegisinceOl write FMuhtevaDegisinceOl;
    // ...
```

THafiza sınıfının "MuhtevaDegisinceOl" olayı simülasyon sisteminin ana penceresini oluşturan TfrmBB3 sınıfında tanımlı "MuhtevaDegisti" adlı yöntem ile işlenir. "MuhtevaDegisinceOl" olayı ve diğer bazı olaylar için yöntem – olay işleyici arasındaki bağlantı aşağıdaki şekilde kurulur. Eşitliğin sol tarafında ilgili nesne ve olayın adı sağ tarafında ise o olayı işleyecek olan olay işleyici yöntemin adı yer almaktadır.

```
// ...
Bilgisayar.Hafiza.MuhtevaDegisinceOl := MuhtevaDegisti;
Bilgisayar.BaslamadanOnceOl := BilgisayarBaslayacak;
Bilgisayar.BasladiktanSonraOl := BilgisayarBasladi;
Bilgisayar.DuruncaOl := BilgisayarDurdu;
Bilgisayar.YaziciYazdiktanSonraOl := BilgisayarCiktiGonderildi;
Bilgisayar.Islemci.SaatdenOnceOl := BilgisayarSaatdenOnce;
Bilgisayar.Islemci.SaatdenSonraOl := BilgisayarSaatdenSonra;
Bilgisayar.Islemci.MikroIslemYapilincaOl :=
  BilgisayarMikroIslemYapildi;
// ...
```

TYazici sınıfı çıkış simülasyonundan sorumludur. TTimer nesnesinden türetilmiştir. TTimer nesnesi belirli zaman aralıkları ile bir *OnTimer* olayını tetikler ve ilgili olay işleyicisinde istenen işlemler yapılabilir. TYazici sınıfında *OnTimer* olayının işlenmesi için, zaman aralığının belirlendiği *Interval* özelliği 1000 mili saniye olarak ayarlanmıştır. İstenirse kullanıcı bu süreyi değiştirebilir. TYazici sınıfının bir yöntemi ve iki özelliği vardır. Bunlar; yazıcının bir şekilcik yazmasını sağlayan *Yaz* yöntemi, yazıcının çıktısını görmemizi sağlayan *Kagit* özelliği ve yazıcı bir şekilciği yazdıktan sonra meydana gelen *YazilincaOl* olaylarıdır. *Kagit* özelliği, TMemo örneği bir nesne ile eşleştirildiğinde yazıcının çıktısı bu nesne üzerinden görülebilir. *YazilincaOl* olayı TBilgisayar sınıfı içinde işlenir. Burada, yazıcı bir şekilciği yazdıktan sonra BB/3'ün çıkış bayrağı (ÇIKB) bir yapılır.

4.2. BBSD Birleştirici Modülü

Bizim bilgisayar simgesel dil birleştiricisine ait sınıfları içerir. Bu sınıflar TBBSDIzlence, TBBSDParca ve TBBSDParcalayici sınıflarıdır.

TBBSDIzlence sınıfının görevi, BBSD'de yazılan izlenceleri makine diline çevirmektir. Bunun yanında izlence yazımı sırasında; tanımlanan simgeler ve yapılan yazım hataları hakkında sisteme, olaylar aracılığı ile bilgi verir.

TBBSDIzlence sınıfı TStringList sınıfından türetilmiştir. TStringList sınıfı bir dizi dizgiyi depolamak ve üzerinde değişiklikler yapmak için kullanılabir [10].

TBBSDIzlence sınıfı, simgesel dildeki izlenceyi makine diline çevirmek için ilk önce izlenceyi oluşturan dizgiyi parçalara (atomlara) ayırır. Bu amaçla TBBSDParcalayici sınıfından faydalanır. TBBSDParcalayici sınıfı bir genelleyici (generic) sınıf olan TObjecList sınıfından türetilmiştir. TObjecList, nesnelerine bir indisle ulaşılabilen sıralı bir nesne listesini temsil eder. Listesinden bir nesne çıkartıldığında onu hafızadan da siler [11].

Genelleyici sınıflar (bazen parametrelendirilmiş türler veya genel türler olarak da anılır) tür güvenliğinden ödün vermeden daha genel kapsamlı kodların yazımına izin verir [12]. Bu sınıfların özelliği belirli bir türdeki veri yerine, farklı türlerde ki verilerle kullanulabilmeleridir. Örneğin "Generics.Collections" biriminde tanımlı olan "TList" sınıfından türetilen bir nesne ile istenirse tamsayılardan veya dizgilerden oluşan bir liste yönetilebilir. Yapılması gereken tek şey ilgili nesnenin tanımında hangi tür veri ile kullanılacağının belirtilmesidir. Böylesi bir tanım aşağıdaki gibi yapılabilir.

```
var
  Tamsayilar: TList<Integer>;
  Dizgiler: TList<string>;
//...
```

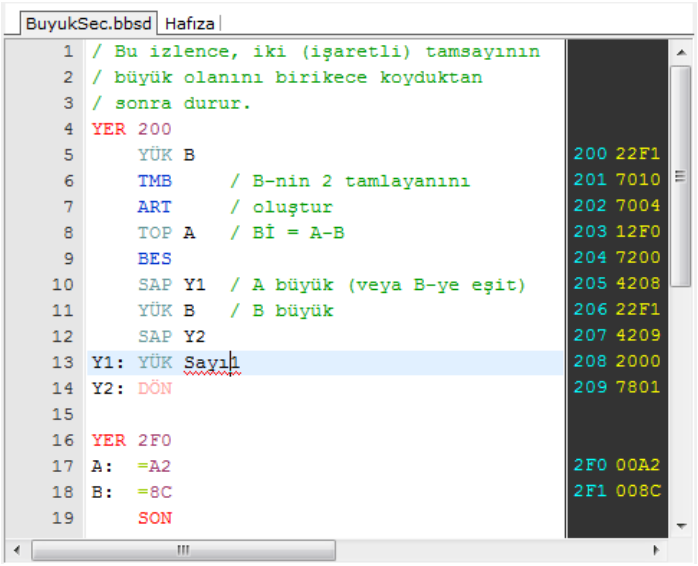
TBBSDParcalayici sınıfı izlenceyi oluşturan satırlardan sadece birini parçalarına (atomlarına) ayırır ve her bir parçayı bir TBBSDParca sınıfı örneği bir nesne içine yerleştirerek saklar. İzlence içindeki herhangi bir satır değiştiğinde sadece o satırla ilişkilendirilmiş TBBSDParcalayici nesnesi yeniden ilgili satırı parçalarına ayırır. Bütün bir izlencenin tekrar baştan sona parçalarına ayrılmasına gerek yoktur. TBBSDParcalayici ilgili satırı parçalarına ayrıştırdıktan sonra eğer parça Çizelge 2.11'de yer alan BBSD komutlarından birisi ise TBBSDParca sınıfının *Kod* özelliği burada hemen atanır. Kod özelliği, izlencenin makine diline çevrilişinde kullanılacak veriyi tutar. Örneğin parça değeri YER gibi bir yardımcı komut vaya bir açıklayış ise bunlar için kod değeri olarak sıfır atanır. Eğer parça değeri YVK ise kod özelliğinin değeri 7840_{16} olarak belirlenir. TBBSDParcalayici sınıfı birleştiricinin işini kolaylaştırmak için boşlukları, yorumları, işleçleri, yardımcı komutları ve sabit

simgelerden (bkz. Çizelge 2.11) olmayan herşeyi işaretler. Bu amaçla TBBSDParca sınıfının *Tur* özelliğini kullanır.

TBBSDIzlence sınıfı yazım hataları konusunda kullanıcıyı bilgilendirmenin yanı sıra bazı anlamsal hatalarla ilgili olarak ta kullanıcıyı uyarır. Örneğin dolaylı adresleyiş sadece hafıza adresleyen komutlar için geçerlidir. Eğer kullanıcı izlence içinde "D BSS" gibi bir kod yazarsa "Hafıza adresleyebilen bir komut bekleniyordu." şeklinde bir mesajla kullanıcıyı uyarır.

4.3. BBSD Düzenleyici Modülü

Bizim bilgisayar simgesel dilinde izlence yazmak için geliştirilen bileşene ait sınıfları içerir. Bu sınıflar TBBSDDuzenleyici, TBBSDDKaydirmaCubuklari ve TBBSDDRenkler sınıflarıdır.



Resim 4.1. BBSD düzenleyicinin ekran görüntüsü

Delphi bileşenleri, Delphi tümleşik geliştirme ortamında yer alan tasarım araçları ile yönetilebilen özel nesnelerdir.

Yeni bir bileşen oluşturmak için genellikle mevcut bir bileşenden yola çıkarsınız. Her iki durumda da bileşeniniz, bileşen hiyerarşisinin temel üç sınıfından birinden türetilir [9]. Bunlar; bir pencereye dayalı olan, giriş odağını ve işletim sistemi iletilerini alabilen TWinControl sınıfı, işletim sistemi kaynaklarının kullanımından tasarruf sağlayan giriş odağını alamayan ve işletim sistemi iletilerine doğrudan cevap veremeyen TGraphicControl sınıfı ve bütün bileşenlerin ebeveyn sınıfı oluşunun yanında görsel olmayan bileşenlerin ebeveyni olarak kullanılabilecek TComponent sınıflarıdır.

TBBSDDuzenleyici sınıfı, klavye ve fare ile etkileşim içinde olan görsel bir bileşen olduğu için TWinControl halefi TCustomControl sınıfından türetilmiştir. TCustomControl sınıfı kendi görüntüsünü oluşturabilmek için yardımcı nesnelere sahiptir.

Bir işletim sistemi iletisi, işletim sistemi tarafından izlenceyi belirli bir olayın oluşu hakkında bilgilendirmek için işletim sistemi tarafından izlenceye gönderilir. Fare ile bir yere tıklandığında, bir pencerenin boyutu değiştirildiğinde veya klavyeden bir tuşa basıldığında Windows, izlenceye bir ileti göndererek bunu bildirir [13].

TBBSDDuzenleyici sınıfı tarafından işlenen işletim sistemi iletileri Çizelge 4.1'de listelenmiştir.

Çizelge 4.1. TBBSDDuzenleyici bileşeni tarafından işlenen işletim sistemi iletileri

İleti Sabiti	Açıklanışı
WM_SETFOCUS	Bir pencereye, klavye odağını aldığında gönderilir[14].
WM_KILLFOCUS	Bir pencereye, klavye odağını kaybettiğinde (başka bir bileşene devrettiğinde) gönderilir [15].
WM_SIZE	Bir pencereye boyutu değiştirildikten sonra gönderilir [16].
WM_GETDLGCODE	Varsayılan olarak, bir denetim üzerindeki bütün klavye girdilerini sistem ele alır; sistem belirli klavye girdilerini, iletişim kutusu gezinti tuşları olarak yorumlar. Bu varsayılan davranışı değiştirmek için denetim WM_GETDLGCODE iletisine cevap verir ve bu tür girdileri kendisi işlemek istediğini belirtir [17].
WM_VSCROLL	Pencerenin dikey kaydırma çubuğunda, bir kayma olayı gerçekleştiği anda pencereye gönderilir [18].
WM_HSCROLL	Pencerenin yatay kaydırma çubuğunda, bir kayma olayı gerçekleştiği anda pencereye gönderilir [19].

Bazı işletim sistemi iletileri doğrudan işlenmeyip, selef sınıfta tanımlı o iletiyi işleyen sanal yöntemler yeniden tanımlanarak işlenmiştir. Bu amaçla; Paint, KeyDown, KeyPress ve MouseDown yöntemleri yeniden tanımlanmıştır.

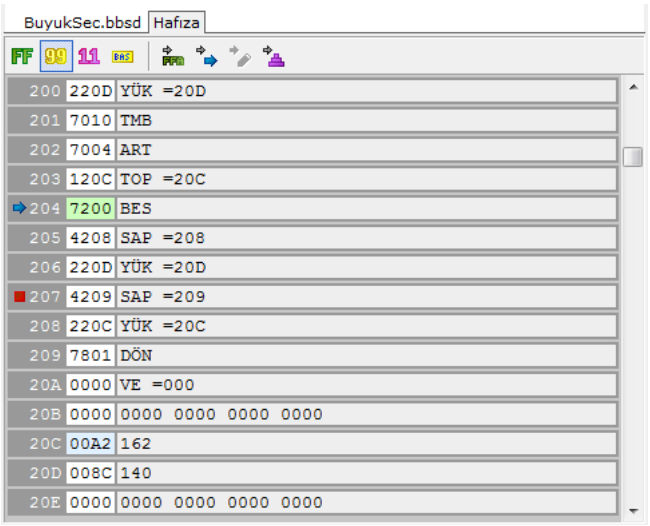
Düzenleyici; kod okunabilirliğini artırmak için yazım renklendiriş (Syntax Highlighting) özelliğini destekler. Yazım hatalarını vurgulamak için hatalı yazılan öğenin altını çizerek gösterir. İzlence yürütümü esnasında (bilgisayar çalışıyorken) hangi satırdaki komut ya da komutların yürütüldüğünü gösterir. İzlencedeki her bir satırı, satır numaraları ve eğer varsa makine dili karşılıkları ile birlikte gösterir. Simgesel dilde yazılmış izlence üzerinde durak noktalarının tanımlanışına imkân verir.

BBSD Düzenleyici nesnesinin sol kenarında yer alan;  simgesi bilgisayar çalışırken hangi satırdaki komutun ifa edildiğini,  simgesi o satırda bir durak noktası tanımlandığını gösterir.

4.4. Hafıza Düzenleyici Modülü

Hafıza muhtevasını görüntülemek, izlemek ve düzenlemek için geliştirilen bileşene ait sınıfları içerir. Bu sınıflar THafizaDuzenleyici ve THDRenkleri sınıflarıdır.

Hafıza düzenleyici üç sütunlu bir cetvel şeklinde tasarlanmıştır. Birinci sütün da adres, ikinci sütun da muhteva ve üçüncü sütunda yorum yer alır. Adres ve muhteva onaltılık tabanda gösterilir. Yorumun ise ne şekilde gösterileceğini kullanıcı belirler. Hafıza düzenleyici bir hafıza hücresindeki muhtevayı dört farklı şekilde yorumlayabilir. Bunlar, 16'lık tabanda sayı, 10'luk tabanda işaretli tamsayı, 2'lik tabanda sayı ve simgesel dil karşılığı şeklindedir. Aynı zamanda İzlence yürütümü esnasında (bilgisayar çalışıyorken) ifa edilen komutun yerini, yığıt için kullanılan hafıza hücrelerini ve muhtevası en son değişen hafıza hücresini gösterebilir. Hafıza düzenleyici üzerinde de tıpkı BBSD düzenleyici üzerinde olduğu gibi durak noktaları tanımlanabilir.



Resim 4.2. Hafıza düzenleyicinin ekran görüntüsü

Hafıza düzenleyicinin hemen üzerinde yer alan araç çubuğunda bulunan düğmelerden ilk dördü bir hafıza hücresindeki muhtevanın nasıl yorumlanacağını belirler. İkinci dörtlü ise hafıza üzerinde gezinmek (belirli bir adresin bulunduğu sayfayı görüntülemek) için kullanılır. Hafıza düzenleyici üzerindeki düğmelerin görevleri Çizelge 4.2'de açıklanmaktadır.

Çizelge 4.2. Hafıza düzenleyici araç çubuğunda yer alan düğmeler ve görevleri

Düğme Simgesi	Görevi
FF	Seçili hafıza adresindeki verinin 16-lık sayı karşılığını göster.
99	Seçili hafıza adresindeki verinin 10-luk sayı karşılığını göster.
11	Seçili hafıza adresindeki verinin 2-lik sayı karşılığını göster.
BAS	Seçili hafıza adresindeki veriyi çözümleyerek komut karşılığını göster.
	İmleci belirtilen hafıza adresine taşı.
	İmleci komut işaretçisinin gösterdiği hafıza adresine taşı.
	İmleci en son değeri değişen adrese taşı.
	İmleci Yığıt işaretçisinin gösterdiği hafıza adresine taşı.

Hafıza düzenleyici nesnesinde adreslerin hemen sol kenarında yer alan; simgesi bilgisayar çalışırken hangi adresteki komutun ifa edildiğini, simgesi o adreste bir

durak noktası tanımlandığını,  simgesi o adresteki muhtevanın değiştiğini ve  simgesi yığıtın hangi adresten itibaren başladığını gösterir.

4.5. Form Modülü

Sistemin görsel öğelerini oluşturan form sınıflarını içerir. Bazı formlar ana pencere içine yuvalanarak gösterilmektedir. Bu formlar, Giriş – Çıkış formu, Gözetleyiciler formu, Kayıtlıklar formu ve Simge Cetveli formlarıdır.

Bir bileşenin *Parent* özelliği, bu bileşenin görüntülenmesinden hangi bileşenin sorumlu olduğunu gösterir [9]. Bir formun içeriğinin (onun ebeveyni olduğu bütün nesnelerin) başka bir form içinde gösterilmesi için ilgili nesnelerin *Parent* özellikleri değiştirilmiştir. Bir formdaki bütün nesnelerin *Parent* özelliklerini ayrı ayrı değiştirmemek için önce formun bütün yüzeyini kaplayan bir Panel nesnesi forma yerleştirilmiş daha sonra asıl bileşenler bu Panel nesnesi içine yerleştirilmiştir. Panel nesnesinin ebeveyni değiştirilerek ilgili formdaki bütün nesneler ana pencereye aktarılmıştır. Örneğin Kayıtlıklar penceresini ana pencerenin sol üst kısmına alan kod aşağıdaki gibidir.

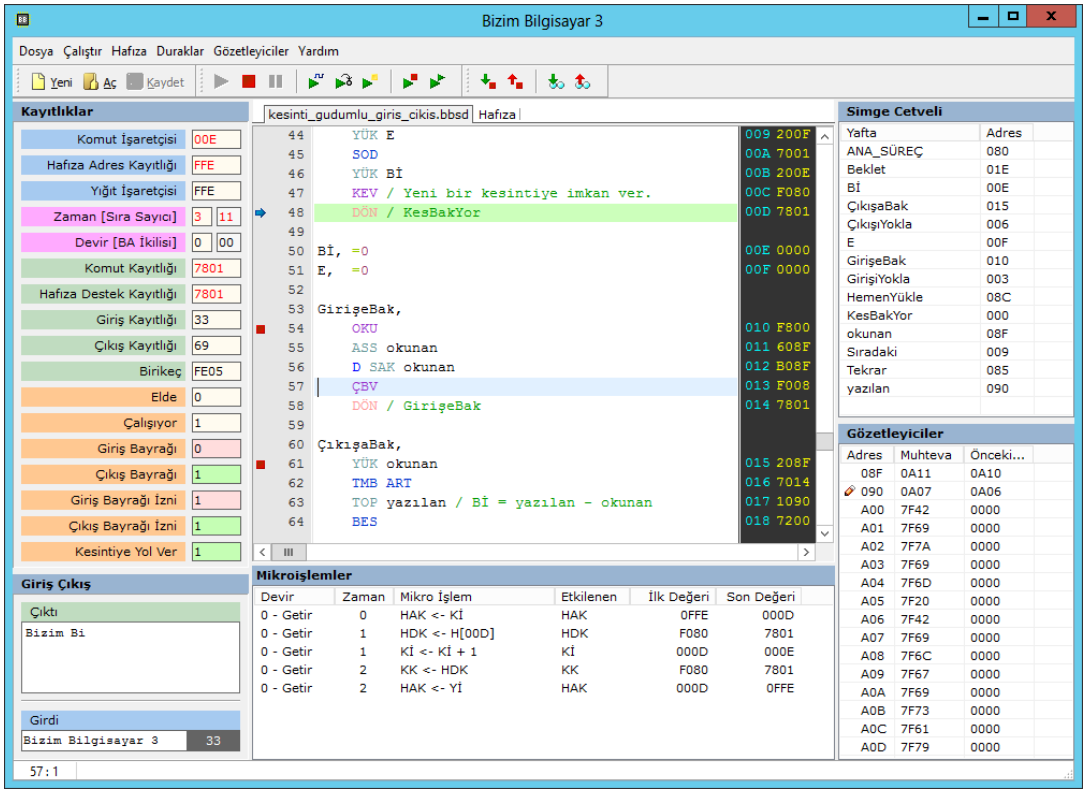
```
// ...
frmKayitliklar.pnlIcerik.Parent := pnlSolUst;
// ...
```

4.6. Birimler Modülü

Genel amaçlı işlevleri ve sınıfları içeren Delphi birimlerini (Unit) içerir.

4.7. BB/3 Simülasyon Sistemi Ekran Öğeleri

BB/3 simülasyon sisteminde temel bileşenler ana pencerede toplanmış, bütün simülasyon sonuçları ana pencereden izlenebilmektedir. Simülasyon sisteminin ekran görüntüsü Resim 4.3'te görülebilir.



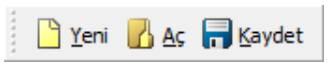
Resim 4.3. BB/3 simülasyon sisteminin ekran görüntüsü

Simülasyon sistemi ana penceresindeki Kayıtlıklar Paneli işlemci kayıtlıklarının muhtevalarını ve muhtevalarındaki değişimleri göstermektedir. Simge Cetveli Paneli simgesel dilde yazılan izlencedeki yaftaları ve adreslerini göstermektedir. Giriş – Çıkış Paneli giriş ve çıkış işlemleri için kullanılabilmektedir. Gözetleyiciler Paneli hafızanın belirli hücrelerindeki değişimi izlemek için kullanılabilmektedir. Simgesel dil düzenleyici sekmesinde Bizim Bilgisayar Simgesel Dilinde izlenceler yazılabilir. Hafıza sekmesinde ise hafıza içeriği görüntülenebilir ve düzenlenebilir. Ana pencerenin orta alt kısmında yer alan panel izlence yazıyorken yazım hatalarını gösterir. Bilgisayar çalışırken ise işlemci tarafından gerçekleştirilen mikro işlemleri gösterir.

Ana pencerede üç araç çubuğu, beş panel ve iki sekme yer almaktadır.

4.7.1. Araç Çubukları

Dosya araç çubuğu: Bu araç çubuğunda yer alan komutlar sırayla; yeni bir izlence oluşturmak için "Yeni", daha önceden yazılmış izlenceleri yüklemek için "Aç" ve düzenlenen izlenceyi kaydetmek için "Kaydet" komutlarıdır.



Resim 4.4. "Dosya" araç çubuğunu

Çalıştır araç çubuğu: Bu araç çubuğunda yer alan komutlar sırayla; bilgisayarı çalıştıran "Çalıştır", bilgisayarı durduran "Durdur", bilgisayar serbest kipte çalışıyorken sistem saat üreticisini durduran "Beklet", bilgisayara bir zaman süresince saat sinyali gönderen "Sonraki Zaman", bilgisayara bir devir tamamlanıncaya kadar saat sinyali gönderen "Sonraki Devir", bilgisayara bir komut tamamlanıncaya kadar saat sinyali gönderen "Sonraki Komut", bilgisayara izlence ya da hafıza üzerinde tanımlanmış ilk durak noktasına ulaşıncaya kadar saat darbesi gönderen "Sonraki Durak" ve bilgisayara sürekli saat sinyali gönderen "Serbest Çalıştır" komutlarıdır.

Resim 4.5. "Çalıştır" araç çubuğunu

Duraklar ve Gözetleyiciler araç çubuğu: Bu araç çubuğunda yer alan komutlar sırayla; Geçerli hafıza hücresi üzerine durak noktası ekleyen "Durak Noktası Ekle", varolan durak noktasını silen "Durak Noktasını Sil", geçerli hafıza hücresini izlemek üzere Gözetleyiciler paneline yeni bir gözetleyici ekleyen "Gözetleyici Ekle" ve geçerli hafıza hücresi ile ilişkilendirilmiş gözetleyiciyi silen "Gözetleyici Sil" komutlarıdır.

Resim 4.6. "Duraklar ve Gözetleyiciler" araç çubuğunu

4.7.2. Paneller

Simge Cetveli Paneli: Simge cetvelinde, izlence içinde tanımlanan yaftalar ve yaftaların adres karşılıkları yer alır. Bu cetvel alfabetik sırada düzenlenmiştir ve büyük küçük harf duyarlılığı yoktur.

Simge Cetveli

Yafta	Adres
ANA_SÜREÇ	080
Beklet	01E
Bİ	00E
ÇıkışaBak	015
ÇıkışıYokla	006
E	00F
GirişeBak	010
GirişiYokla	003

Resim 4.7. "Simge Cetveli" paneli

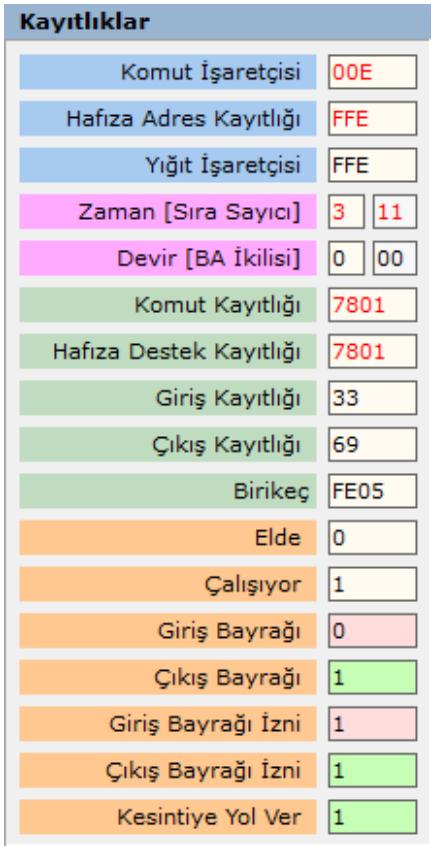
Gözetleyiciler Paneli: Bu panelde gözetleyiciler tarafından izlenen hafıza hücrelerinin adresleri, muhtevaları ve muhtevalarındaki değişimler listelenir. Hafıza hücrelerinin muhtevaları tıpkı hafıza düzenleyicede olduğu gibi dört farklı şekilde sergilenebilir. Bir hücrenin muhtevasının ne şekilde sergileneceği gözetleyici tanımlanırken belirlenebileceği gibi daha sonra bu panel üzerinde fare ile sağ tuşa basılarak açılan menüden de seçilebilir. Bir gözetleyici adresinin sol tarafındaki simgesi, o adresteki muhtevanın son ifa edilen komut dolayısıyla değiştiği anlamına gelir.

Gözetleyiciler

Adres	Muhteva	Önceki...
08F	0A11	0A10
090	0A07	0A06
A00	7F42	0000
A01	7F69	0000
A02	7F7A	0000
A03	7F69	0000

Resim 4.8. "Gözetleyiciler" paneli

Kayıtlıklar Paneli: Kayıtlıklar paneli işlemci içinde ki kayıtlıkların muhtevalarını ve muhtevalarında ki değişimlerin izlenişini sağlar. Eğer bir kayıtlığın muhtevası kırmızı renkte sergileniyorsa bu, o kayıtlığın muhtevasının değiştiği anlamına gelir. Muhtevası değişen bir kayıtlığın bir önceki muhtevası fare ile o kayıtlığın muhtevası üzerine gelindiğinde durum çubuğunda ya da biraz beklenildiğinde açılan ipucu penceresinde görülebilir. Kesinti ile ilgili bayrakların gösteriminde kullanılan pembe arkaplan rengi, kesintiye izin verilmediği, açık yeşil arkaplan rengi ise kesintiye izin verildiği anlamına gelir. Örneğin Resim 4.8'e göre; yazılımsal olarak kesintiye izin verildiği ("Kesintiye Yol Ver" açık yeşil), çıkış birimi kaynaklı bir kesintinin kabul edileceği ("Çıkış Bayrağı" ve "Çıkış Bayrağı İzni" açık yeşil) fakat giriş birimi kaynaklı bir kesintinin kabul edilmeyeceği ("Giriş Bayrağı" ve "Giriş Bayrağı İzni" pembe) bayrakların arkaplan renklerine bakılarak anlaşılabilir.



Resim 4.9. "Kayıtlıklar" paneli

Giriş Çıkış Paneli: Bu panelin girdi bölümünde yazıcı çıktısını sergilemek amacı ile bir TMemo bileşeni kullanılmıştır. Bu bileşen salt okunur olarak ayarlanmıştır. Düzenleme adına üzerinde yapılabilecek tek işlem içeriğinin tamamen silinmesidir. Bu da, üzerine fare ile çift tıklayarak yapılabilir. Girdi bölümünde ise bir TEdit bileşeni kullanılmıştır. BB/3'e klavye ile girdi göndermek istenirse bu bileşene tıklanmalı ve simülasyon sisteminin üzerinde çalıştığı gerçek bilgisayarın klavyesi kullanılmalıdır. Klavye üzerinde bir tuşa basıldığında onunla ilişkilendirilmiş şekilciğin ASCII kodu giriş kayıtlığına yazılır. BB/3'e girilen son 20 şekilcik bu metin kutusunda görülebilir. 21. şekilcik girildiğinde veya enter tuşuna basıldığında metin kutusu temizlenerek yeniden dolmaya başlar. Eğer BB/3'ün giriş kayıtlığı dolu ise (GİRB=1 ise) bir bip sesi duyulur ve girdi kabul edilmez.

Resim 4.10. "Giriş Çıkış" paneli

Mikroişlemler Paneli: Bu panel bilgisayar çalışıyorken görüntülenir. Mikroişlemler panelinde bir komutun ifası tamamlanıncaya kadar, hangi devir ve zamanda hangi mikroişlem yada mikroişlemlerin yapıldığı, bu işlemlerden hangi kayıtlıkların ne şekilde etkilendiği (önceki ve sonraki değeri) listelenir. Yeni bir komutun ifası başladığında panel içeriği temizlenir ve o komuta ait mikroişlemler listelenir.

İletiler paneli: Bu panel bilgisayar çalışmıyorken görüntülenir ve yazılan BBSD izlencesindeki yazım hatalarını, hatanın konumu (satır, sütun) ile birlikte listeler.

Mikroişlemler

Devir	Zaman	Mikro İşlem	Etkilenen	İlk Değeri	Son Değeri
0 - Getir	0	HAK <- Kİ	HAK	0FFE	000D
0 - Getir	1	HDK <- H[00D]	HDK	F080	7801
0 - Getir	1	Kİ <- Kİ + 1	Kİ	000D	000E
0 - Getir	2	KK <- HDK	KK	F080	7801
0 - Getir	2	HAK <- Yİ	HAK	000D	0FFE

Resim 4.11. “Mikroişlemler” paneli

İletiler

11-5 : Devamında bir adres yada yafta bekleniyordu.
11-9 : Bilinmeyen simge.
11-13 : Bilinmeyen simge.

Resim 4.12. “İletiler” paneli

5. DENEYLER

Geliştirilen simülasyon sisteminin çalışmasını test etmek amacıyla çeşitli deneyler yapılmıştır. Her bir deney, genelde bütün sistemi sınamak için yapılmıştır. Özelde ise sistemin belirli bir özelliğine odaklanmaktadır.

Deney 1: Aşağıda hafızadaki iki sayıdan büyük olanını birikece yükledikten sonra duran bir izlence yer almaktadır[1]. Bu izlence ile Simge Cetveli panelinin ve Kayıtlıklar panelinin işlerliği sınanmıştır.

```
/ Bu izlence, iki (işaretli) tamsayının büyük olanını birikece koyduktan
/ sonra durur.
YER 200
    YÜK B
    TMB     / B'nin 2-tamlayanını
    ART     / oluştur.
    TOP A   / Bİ = A-B
    BES
    SAP Y1  / A büyük (veya B'ye eşit)
    YÜK B   / B büyük
    SAP Y2
Y1: YÜK A
Y2: DÖN

YER 2F0
A:  =A2
B:  =8C
    SON
```

Deney 2: Aşağıda hafızadaki iki sayıyı topladıktan sonra sonucu yine bir hafıza hücresinde saklayan bir izlence yer almaktadır. Bu izlence ile Gözcüler panelinin işlerliğini sınanmıştır.

```
YER 200
    YÜK =375
    TOP =376
    SAK =377
    DÖN

YER 375
    =96
    =77
    =0

    SON
```

Deney 3: Aşağıda girişten okuduğu 10 adet şekilciği çıkış birimine yazdırdıktan sonra duran bir izlence yer almaktadır [1]. Bu izlencede giriş – çıkış izlence güdümlü olarak takip edilmektedir. Bu izlence ile Giriş – Çıkış panelinin işlerliği sınanmıştır.

```
YER 100
YUKARI:
    DÜS OKUYAZ
    ASS SAYAÇ
    SAP YUKARI
    DÖN

YER 200
OKUBAYT:
    GBS
    SAP OKUBAYT
    OKU
    DÖN

YAZBAYT:
    ÇBS
    SAP YAZBAYT
    YAZ
    DÖN

OKUYAZ: / Yankılı okuyuş
    DÜS OKUBAYT
    DÜS YAZBAYT
    DÖN

YER 300
    SAYAÇ: = -A
    SON
```

Deney 4: Aşağıda girişten 10 şekilcik okuyarak bunları çıkış birimine yazdıran bir izlence yer almaktadır[1]. Bu izlence ile Giriş – Çıkış simülasyonunun doğruluğu sınanmıştır. Aynı zamanda kesinti güdümlü giriş çıkışta sınanmıştır.

```
YER 80
ANA_SÜREÇ,
    DÜS HEMENYÜKLE
    =9FF / OKU-YAZ destek-alanı başlangıç adresi öncesi.
    SAK OKUNAN
    SAK YAZILAN
    KEV
TEKRAR,
    DÜS HEMENYÜKLE
    =-BFF
    TOP YAZILAN
    BAS
    SAP TEKRAR
    KEY
        / bu noktada istenen başka işlemler varsa yapılabilir.
    DÖN / izlencenin mantıksal sonu.
HEMENYÜKLE,
    YDY
    Yİİ
    DÖN

OKUNAN:     = 0
YAZILAN:    = 0

/ Kesintiye bakan yordam
YER 000 / Hafızanın ilk yeri
KESBAKYOR:
    SAK Bİ / İşlem kayıtlıklarını kurtar.
    SAD
    SAK E

/ Kesinti sebebini ara:
GİRİŞİYOKLA,
    GBS
    SAP ÇIKIŞIYOKLA
    DÜS GİRİŞEBAK
ÇIKIŞIYOKLA,
    ÇBS
    SAP SIRADAKİ
    DÜS ÇIKIŞABAK

/ Bütün sebepler tarandığından sırada geri dönmek var
SIRADAKİ,
    YÜK E
    SOD
    YÜK Bİ
    KEV / Yeni bir kesintiye imkan ver.
```

```
    DÖN / KESBAKYOR

Bİ, =0
E,  =0

GİRİŞEBAK,
    OKU
    ASS OKUNAN
    D SAK OKUNAN
    ÇBV
    DÖN / GİRİŞEBAK

ÇIKIŞABAK,
    YÜK OKUNAN
    TMB ART
    TOP YAZILAN / Bİ = YAZILAN - OKUNAN
    BES
    SAP BEKLET
    ASS YAZILAN
    D YÜK YAZILAN
    YAZ
    BAS BES

BEKLET,
    ÇBY
    DÖN / ÇIKIŞABAK

    SON
```

6. SONUÇ ve ÖNERİLER

BB/3 simülasyon sistemi bir öğretim dönemi boyunca 37 kişilik bir öğrenci grubunda denenmiştir. Öğrenci grubu BB/3 simülatör ve birleştiricisini kullanarak çeşitli deneyler gerçekleştirmişlerdir. Öğrencilere görüş ve önerileri sorularak simülasyon sistemi üzerinde değişiklikler yapılmıştır. Bu simülasyon sistemini kullanarak öğrencilerin son derece motive edildikleri ve konuları daha kolay kavradıkları gözlemlenmiştir.

Simülasyon sistemi açık kaynak kodludur, ve günümüz uygulama geliştirme teknikleri esas alınarak geliştirilmiştir. Bu simülatörü geliştirmek isteyecekler açısından yazılımın sistematik olmasına dikkat edilmiştir. Sistemin açık kaynak kodlu olarak geliştirilişinden dolayı öğrenciler simülasyon sistemi kodlarını inceleyip anlayabilirler ve hatta donanım üzerinde yapmak istedikleri değişiklikleri Simülasyon sistemine uyarlayıp, simülasyon sistemini değiştirebilirler.

Simülasyon sisteminin için geliştirilen iki bileşen sayesinde, ileri izlence yazım tekniklerinden de faydalanılmış bu konularda deneyim kazanılmıştır.

Simülasyon sistemi geliştirilirken Delphi dilindeki son yeniliklerden de faydalanılmaya çalışılmıştır. Bu özellikler "Genelleyici Sınıflar" ve "Sınıf Yardımcıları" özellikleridir.

Elde edilen kazanımlar özetle aşağıdaki gibi sıralanabilir.

1. Donanım tasarımı ilkeleri hususunda bilgi ve deneyim kazanılmıştır.
2. Sayısal elektronik devrelerin izlencelerle modellenmesi hususunda bilgi ve deneyim kazanılmıştır.
3. Simülatör ileri izlence yazım teknikleri kullanılarak geliştirilmiş ve bu hususlarda deneyim kazanılmıştır.
4. Öğrencilerden alınan olumlu eleştiriler göstermiştir ki yeni simülasyon sistemi BB/3'ün tasarım ve işleyiş esaslarının anlaşılmasında öğrencilere yardımcı olmuştur.

Yeni simülasyon sistemi şu ana kadar geliştirilen sistemlerin en iyisi olmakla birlikte yine de eksik olan kısımları bulunmaktadır. Sistemin daha da ileriye götürülebilmesi adına yapılabilecekleri şu şekilde sıralayabiliriz.

1. Kullanım kolaylığı açısından
    a. BBSD düzenleyicide; kes, kopyala ve yapıştır gibi pano işlemleri desteği eklenebilir.
    b. *.bbsd ve *.bsd uzantılı dosyalar simülatör ile eşleştirilebilir.
    c. Ana penceredeki paneller kaygan panellere dönüştürülebilir.
    d. "Gerçek zamanlı çalıştır" seçeneği eklenebilir.
    e. BBSD Düzenleyici ve Hafıza Düzenleyici renklerinin ve yazı tiplerinin bir iletişim kutusu üzerinden kullanıcı tarafından ayarlanışı sağlanabilir.
    f. Serbest çalışma kipinde, saat darbesi için belirlenen ve yazıcı için belirlenen gecikim sürelerinin bir iletişim kutusu üzerinden kullanıcı tarafından ayarlanışı sağlanabilir.
2. Eğitsel amaçlar açısından
    a. Gözcüler ve hafıza düzenleyici için muhtevanın şekilcik olarak gösterimi
    b. Yardım menüsünde; kullanım kılavuzu, BB/3 ile ilgili belgeler, örnek deneyler yer alabilir.
    c. Giriş – Çıkış paneline çıktıdaki her şekilciğin onaltılık gösterimi için yeni sekme eklenebilir.
    d. BB/3 bilgisayar modeli denetim mantığı ilkesel görünümü üzerinden verilerin akışı grafiksel olarak gösterilebilir.

KAYNAKLAR

1. İnternet: Gazi Üniversitesi "Donanım Tasarımı Dersi, Ders Notları" http://w3.gazi.edu.tr/~dcalik/=Duyuru=/Duyuru=DERSLER/duyuru-Lisans-BiL210/d-BiL210.htm (2012).

2. Çalıkoğlu, D., "Bilgisayar "Donanım işleyiş ilkelerinin öğretiminde etkin bir yaklaşım: Bizim Bilgisayar", **Gazi Üniversitesi Endüstriyel Sanatlar Eğitim Fakültesi Dergisi**, 10(10): 1-11 (2002).

3. İnternet: Iowa Üniversitesi Bilgisayar Bilimleri Bölümü "Doug Jones's DEC PDP-8 Index", http://www.divms.uiowa.edu/~jones/pdp8/ (2012)

4. Mano, M.M., "Bilgisayar Sistemleri Mimarisi", Marşoğlu, A., Suçsuz, N., **Literatür Yayıncılık**, İstanbul, 89, 90, 232 (2002).

5. Çakır, Ö., "Bir birleştiricinin tasarımı ve gerçekleştirilmesi", Yüksek Lisans Tezi, **Gazi Üniversitesi Fen Bilimleri Enstitüsü**, Ankara, 38 (2003).

6. Çetin, G., "Eğitsel bir bilgisayar için donanım simülatörü ve simgesel dil birleştiricisi", Yüksek Lisans Tezi, **Gazi Üniversitesi Bilişim Enstitüsü**, Ankara, (2009).

7. İnternet: National Museum of American History "Perspectives of the Smithsonian: Smithsonian Computer History" http://americanhistory.si.edu/collections/comphist/objects/pdp8.htm (2012).

8. İnternet: Embarcadero Developer Network "Object Pascal Style Guide" http://edn.embarcadero.com/article/10280 (2012)

9. Cantû, M., "Delphi 7 Uygulama Geliştirme Klavuzu", Yağcı, D., Erdem T. L., **Alfa Yayınları**, İstanbul, 171, 325, 367 (2003).

10. İnternet: Embarcadero Product Documentation Wikis "System.Classes.TStringList - XE2 API Documentation" http://docwiki.embarcadero.com/Libraries/en/System.Classes.TStringList (2012).

11. İnternet: Embarcadero Product Documentation Wikis "System.Generics.Collections.TObjectList - XE2 API Documentation" http://docwiki.embarcadero.com/Libraries/en/System.Generics.Collections.TObjectList (2012).

12. İnternet: Felix Colibri- Delphi Source Code, Technical Articles, Delphi Trainings, Delphi Consulting and Development "Delphi .Net Generics Tutorial" http://www.felix-colibri.com/papers/oop_components/delphi_generics_tutorial/delphi_generics_tutorial.html (2012).

13. Teixeira S, Pacheco X., "Borland® Delphi™ 6 Developer's Guide", **Sams Publishing**, Indianapolis, 130 (2001).

14. İnternet: Windows Desktop Development "WM_SETFOCUS message" http://msdn.microsoft.com/en-us/library/windows/desktop/ms646283%28v=vs.85%29.aspx (2012).

15. İnternet: Windows Desktop Development "WM_KILLFOCUS message" http://msdn.microsoft.com/en-us/library/windows/desktop/ms646282%28v=vs.85%29.aspx (2012).

16. İnternet: Windows Desktop Development "WM_SIZE message" http://msdn.microsoft.com/en-us/library/windows/desktop/ms632646%28v=vs.85%29.aspx (2012).

17. İnternet: Windows Desktop Development "WM_GETDLGCODE message" http://msdn.microsoft.com/en-us/library/windows/desktop/ms645425%28v=vs.85%29.aspx (2012).

18. İnternet: Windows Desktop Development "WM_VSCROLL message" http://msdn.microsoft.com/en-us/library/windows/desktop/bb787577%28v=vs.85%29.aspx (2012)

19. İnternet: Windows Desktop Development "WM_HSCROLL message" http://msdn.microsoft.com/en-us/library/windows/desktop/bb787575%28v=vs.85%29.aspx (2012).

EKLER

Ek – 1. BB/3 simülasyon sistemi kullanım klavuzu

BB/3 SİMÜLASYON SİSTEMİ

Dosya İşlemleri

Yeni bir izlence oluşturmak İçin...

Yeni bir izlence oluşturmak için simülatörü çalıştırdığınızda BBSD düzenleyicisi yeni bir izlence yazmanız için hazırdır. Ancak mevcut bir izlencenin ardından yeni bir izlence yazmak isterseniz; Dosya menüsünden Yeni (Ctrl + N) komutunu verin. Daha önceki BBSD izlencenize ait kaydedilmemiş bir değişiklik mevcutsa bu hususta uyarılacaksınız. Ancak hafıza düzenleyici ile hafızada yaptığınız değişiklikler hususunda bir uyarı iletisi almazsınız. Yeni bir izlence oluşturduğunuzda hafıza da sıfırlanacaktır (bütün muhtevalar 0000 olacaktır).

Daha önce kaydettiğiniz bir izlenceyi yüklemek için...

Daha önce kaydettiğiniz bir izlenceyi yüklemek için Dosya menüsünden Aç (Ctrl + O) komutunu verin. Açılan iletişim penceresinden izlencenizin bulunduğu dizini ve ardından izlence dosyasını seçtikten sonra Aç düğmesine tıklayın.

Yazdığınız izlenceyi kaydetmek için...

Yazdığınız izlenceyi kaydetmek için Dosya menüsünden Kaydet (Ctrl + S) komutunu verin. Eğer izlenceyi ilk kez kaydediyorsanız karşınıza izlenceniz için bir dizin ve dosya adı belirlemenizi isteyen Kaydet iletişim kutusu gelecektir. İzlencenin kaydedileceği dizini seçip dosya adını belirledikten Kaydet düğmesine tıklayın.

Ek – 1. (Devam) BB/3 simülasyon sistemi kullanım klavuzu

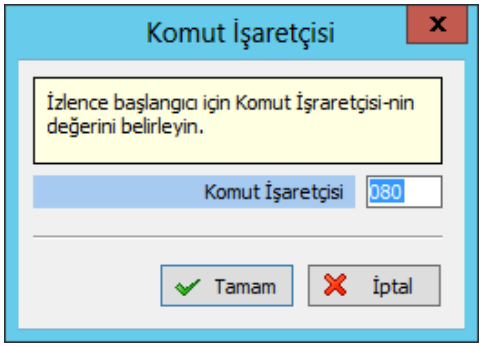
İzlenceyi farklı bir konuma veya farklı bir adla kaydetmek için...

İzlenceyi farklı bir konuma veya farklı bir adla kaydetmek için Dosya menüsünden Farklı Kaydet komutunu verin.

İzlencenin Yürütülmesi

Bilgisayarı çalıştırmak için Çalıştır menüsünden Başlat (F9) komutunu verin.

BBSD düzenleyici etkin iken yazdığınız izlencede bir söz dizim hatası yoksa karşınıza komut işaretçisinin başlangıç değerini belirlemeniz için bir iletişim kutusu gelir.



Resim 1.1. Komut İşaretçisi seçim iletişim kutusu.

Bu pencerede varsayılan olarak BBSD izlencenizde ki ilk hafıza ile ilişkili satırın, ilişkili olduğu adresin değeri seçili gelir. Genelde bu istenen durumdur. Eğer hafıza düzenleyici etkinken Çalıştır komutunu verirseniz, BBSD düzenleyicide yazdığınız izlence dikkate alınmaz, yazım hatası olsa dahi bilgisayar çalışacaktır.

Bilgisayarın çalışsa dahi herhangi bir işlem yapılmadığını göreceksiniz. Bunun sebebi sistem saat üreticisinin pasif durumda oluşudur. Sisteme saat sinyali göndermek için Çalıştır menüsündeki Sonraki Zaman (F5), Sonraki Devir (F6), Sonraki Komut

Ek – 1. (Devam) BB/3 simülasyon sistemi kullanım klavuzu

(F7), Sonraki Durak (F8), veya Serbest Çalıştır (Shift + F8) komutlarından birini vermelisiniz.

Sonraki Durak yada Serbest Çalıştır komutlarından birini verdiyseniz, tekrar sistem saat üreticisini durdurmak için Çalıştır menüsünden Beklet (Ctrl + F9) komutunu veriniz.

Durak Noktaları

Durak noktaları hem BBSD düzenleyici üzerinden hem da hafıza düzenleyici üzerinden tanımlanabilir. Bir durak noktası eklemek için Duraklar menüsünden Durak Noktası Ekle (F2) komutunu verin. Bir satırda yada adreste durak noktası tanımlı olduğunu o satırın yada adresin sol tarafındaki simgesinden anlayabilirsiniz. Bir durak noktasını kaldırmak istiyorsanız; BBSD düzenleyici etkinken durak noktasının bulunduğu satıra gelin, hafıza düzenleyici etkinken durak noktasının bulunduğu adrese gelin ve Duraklar menüsünden Durak Noktasını Sil (Shift + F2) komutunu verin.

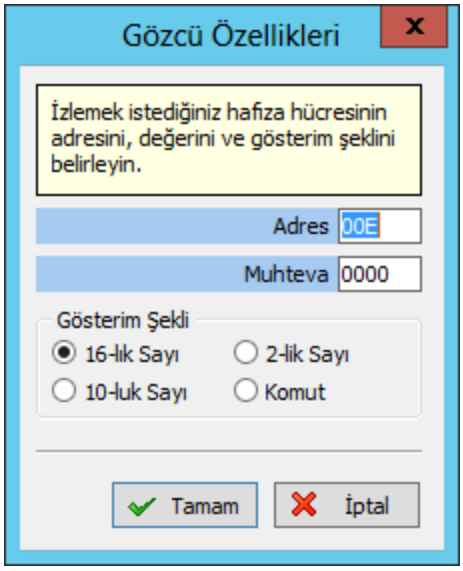
BBSD düzenleyicide satır numaralarına yada hafıza düzenleyicide adreslere tıkladığınızda; eğer o satırda yada adreste ir durak noktası yoksa, o satıra yada adrese bir durak noktası koyarsınız, eğer durak noktası varsa, o durak durak noktasını kaldırırsınız.

Tanımlı bütün durak noktalarını kaldırmak için Duraklar menüsünden Bütün Durak Noktalarını Sil komutunu verin. Bu komut sadece etkin düzenleyici üzerinde tanımlı durak noktalarını siler. Yani hafıza düzenleyici etkinken bu komutu verirseniz, BBSD düzenleyici üzerinde tanımladığınız durak noktalarını da silmiş olmazsınız. Tam tersi de geçerlidir.

Ek – 1. (Devam) BB/3 simülasyon sistemi kullanım klavuzu

Gözetleyiciler

Gözetleyiciler hafıza hücrelerindeki muhteva değişimlerini izlemenize yardımcı olurlar. Bir gözetleyici oluşturmak için Gözetleyiciler menüsünden Gözetleyici Ekle (F3) komutunu verin yada Gözetleyiciler paneli üzerinde fare ile sağ tuşa tıklayarak Ekle komutunu verin. Gözetleyici Özellikleri iletişim kutusu açılacaktır. Bu kutu aracılığı ile izlemek istediğiniz adresi ve muhtevasının ne şekilde sergilenmesini (16'lık tabanda, 10'luk tabanda işaretli tamsayı, 2'lik tabanda veya komut hatırası olarak) istediğinizi belirtebilirsiniz. Eğer isterseniz ilgili adresin değerini de buradan değiştirebilirsiniz.



Resim 1.2. Gözetleyici Özellikleri iletişim kutusu

Eğer bir gözetleyicinin sergileniş şeklini değiştirmek isterseniz, Gözetleyiciler panelinde, o gözetleyici üzerinde farenin sağ tuşuna basın ve açılan menüden istediğiniz sergileniş şeklini seçin.

Bir gözetleyiciyi silmek için; gözetleyicinin izlediği hafıza hücresi ili ilişkili satıra BBSD düzenleyici üzerinden gelin yada ilgili adresi hafıza düzenleyici üzerinde seçin

Ek – 1. (Devam) BB/3 Simülasyon Sistemi Kullanım Klavuzu

ve Gözetleyiciler menüsünden Gözetleyiciyi Sil (Shift + F3) komutunu verin. Kullanabileceğiniz bir diğer yöntem ise Gözetleyiciler panelinden, silmek istediğiniz gözetleyici üzerinde farenin sağ tuşuna basmak ve açılan menüden Sil komutunu vermektir.

Girdi ve Çıktı İşlemleri

BB/3'e girdi göndermek için simülasyon sistemini çalıştırdığınız bilgisayarın klavyesini kullanabilirsiniz. Bunun için Giriş Çıkış panelindeki Girdi metin kutusuna tıklayın ve klavyeden herhangi bir tuşa basın. Bastığınız tuşa ait şekilciğin ASCII kodu GİRB = 1 ise giriş kayıtlığına aktarılacaktır. Ayrıca bu metin kutusunda son girdiğiniz 20 şekilciği de görebilirsiniz. 21. şekilciği girdiğinizde yada enter tuşuna bastığınızda girdi metin kutusu temizlenecektir. Windows Vista ve sonrası işletim sistemleri Unicode kod sistemini kullandığı için bazı şekilcikleri beklediğiniz gibi göremeyebilirsiniz.

BB/3 çıktı birimi olarak yazıcı kullanır ve bu yazıcının çıktısı da simülasyon sisteminde Giriş Çıkış panelinde görüntülenir. Çıktıyı herhangi bir zamanda temizlemek isterseniz (yazıcıya yeni bir kağıt takmak isterseniz) çıktı metin kutusuna farenin sol tuşu ile iki kere tıklayın.

Hafıza Düzenleyici

Hafıza düzenleyici üç sütunlu bir cetvel şeklinde tasarlanmıştır. Birinci sütün da adres, ikinci sütun da muhteva ve üçüncü sütunda yorum yer alır. Adres ve muhteva onaltılık tabanda gösterilir. Yorumun ise ne şekilde gösterileceğini kullanıcı belirler. Hafıza düzenleyici bir hafıza hücresindeki muhtevayı dört farklı şekilde yorumlayabilir. Bunlar, 16'lık tabanda sayı, 10'luk tabanda işaretli tamsayı, 2'lik tabanda sayı ve simgesel dil karşılığı şeklindedir. Bir hücredeki muhtevayı 16'lık tabanda sayı olarak

Ek – 1. (Devam) BB/3 Simülasyon Sistemi Kullanım Klavuzu

görmek için Hafıza menüsünden 16-lık Sayı (Ctrl + 1), 10'luk tabanda işaretli sayı olarak görüntülemek için 10-luk Sayı (Ctrl + 2), 2'lik tabanda sayı olarak görüntülemek için 2-lik Sayı (Ctrl + 3) ve komut karşılığını görmek için Komut (Ctrl + 4) komutlarından birini verin.

Hafıza düzenleyici içinde gezinmek için yukarı ve aşağı ok tuşlarını, home ve end tuşlarını ve page up ve page down tuşlarını kullanabilirsiniz. Bunun yanında belirli bir hafıza adresine gitmek için Hafıza menüsünden Adrese Git (Ctrl + Alt + A) komutunu verin. Açılan iletişim penceresinde Hafıza Adresi metin kutusuna gitmek istediğiniz hafıza adresini yazın ve Tamam düğmesine tıklayın. Hafıza düzenleyici, BBSD düzenleyicide olduğu gibi bilgisayar çalışıyorken komut işaretçisini takip etmez. Eğer komut işaretçisinin gösterdiği adrese gitmek istiyorsanız Hafıza menüsünden Komut İşaretçisine Git (Ctrl + Alt + K) komutunu verin. Eğer muhtevası en son değişen hafıza hücresini görmek istiyorsanız Hafıza menüsünden Değişen Adrese Git (Ctrl + Alt + D) komutunu verin. Eğer yığıtı görüntülemek istiyorsanız Hafıza menüsünden Yığıt İşaretçisine Git (Ctrl + Alt + Y) komutunu verin.

Ek – 2. BB/3 simülasyon sistemi kısayol tuş birleşimleri

Menü Seçeneği	Kısayol Tuş Birleşimi	Açıklanışı
Dosya – Yeni	Ctrl + N	Yeni izlence oluştur.
Dosya – Aç	Ctrl + O	Varolan izlenceyi dosyadan yükle.
Dosya – Kaydet	Ctrl + S	İzlenceyi kaydet.
Dosya – Farklı Kaydet	Ctrl + Shift + S	İzlenceyi yeni bir adla ve/veya yeni bir konuma kaydet.
Dosya – Çıkış	Alt + F4	Simülatörü kapat.
Çalıştır – Başlat	F9	Çalışıyor ikilisini kur. Bilgisayarı çalıştır. Bu durumda izlence üzerinde değişiklik yapılamaz.
Çalıştır – Durdur	Shift + F9	Çalışıyor ikilisini sil. Bilgisayarı durdur.
Çalıştır – Beklet	Ctrl + F9	Bilgisayar serbest modda yada durak noktasına kadar çalışıyorken sistem saat üreticisini durdurur.
Çalıştır – Sonraki Zaman	F5	Bir sonraki zaman tamamlanana kadar işlemciye saat sinyali gönder.
Çalıştır – Sonraki Devir	F6	Bir sonraki devir tamamlanıncaya kadar işlemciye saat sinyali gönder.
Çalıştır – Sonraki Komut	F7	Bir sonraki komut tamamlanıncaya kadar işlemciye saat sinyali gönder.
Çalıştır – Sonraki Durak	F8	Bir sonraki durak noktasına gelinceye kadar işlemciye saat sinyali gönder.
Çalıştır – Serbest Çalıştır	Shift + F8	İşlemciye sürekli saat sinyali gönderir. Komut işaretçisi bir durak noktasına gelse dahi sistem saat üreticisi durmaz.
Hafıza – 16-lık Sayı	Ctrl + 1	Seçili hafıza adresindeki verinin 16-lık sayı karşılığını göster.
Hafıza – 10-luk Sayı	Ctrl + 2	Seçili hafıza adresindeki verinin 10-luk sayı karşılığını (işaretli olarak) göster.
Hafıza – 2-lik Sayı	Ctrl + 3	Seçili hafıza adresindeki verinin 2-lik sayı karşılığını göster.
Hafıza – Komut	Ctrl + 4	Seçili hafıza adresindeki veriyi çözümleyerek komut karşılığını göster.
Hafıza – Adrese Git	Ctrl + Alt + A	İmleci belirtilen hafıza adresine taşı.
Hafıza – Komut İşaretçisine Git	Ctrl + Alt + K	İmleci komut işaretçisinin gösterdiği hafıza adresine taşı.
Hafıza – Değişen Adrese Git	Ctrl + Alt +D	İmleci en son değeri değişen adrese taşı.
Hafıza – Yığıt İşaretçisine Git	Ctrl + Al t +Y	İmleci Yığıt işaretçisinin gösterdiği hafıza adresine taşı.
Gözetleyiciler – Gözetleyici Ekle	F3	Muhteva değişimini izlemek için gözetleyici oluştur.
Gözetleyiciler – Gözetleyiciyi Sil	Shift+ F3	Gözetleyiciyi Sil.
Gözetleyici – Bütün Gözetleyicileri Sil	Ctrl + Shift + F3	Bütün gözetleyicileri sil.
Duraklar – Durak Noktası Ekle	F2	İmlecin bulunduğu satıra durak noktası ekle.
Duraklar – Durak Noktasını Sil	Shift + F2	İmlecin bulunduğu satırda ki durak noktasını sil.
Duraklar – Bütün Durak Noktalarını Sil	Ctrl + Shift + F2	Bütün durak noktalarını iptal eder.

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, Adı	: KARASU, Namık Kemal
Uyruğu	: T.C.
Doğum Tarihi ve Yeri	: 12.09.1981 Kahramanmaraş
Medeni Hali	: Bekar
Telefon	: 0 (507) 956 6692
e–posta	: nkkarasu@nkkarasu.net

Eğitim

Derece	Eğitim Birimi	Mezuniyet Tarihi
Lisans	Gazi Üniversitesi/ ESEF Bilgisayar Öğretmenliği	2003
Lise	Çukurova Elektrik AML/ Bil. Donanımı Bölümü	1999

İş Deneyimi

Yıl	Yer	Görev
2004 – 2007	MEB Beypazarı Meslek Lisesi	Bil. Tek. Öğretmeni
2007 – 2010	MEB Bulancak 19 Eylül KTML	Bil. Tek. Dal Şefi
2010 – ...	MEB Pursaklar İMKB Tek. ve EML	Bil. Tek. Öğretmeni

Yabancı Dil

İngilizce

Hobiler

Elektronik devreler, Futbol, Sinema