EĞİTSEL BİR BİLGİSAYAR İÇİN
DONANIM SİMÜLATÖRÜ VE SİMGESEL DİL BİRLEŞTİRİCİSİ

Gökmen ÇETİN

YÜKSEK LİSANS TEZİ
BİLGİSAYAR EĞİTİMİ

GAZİ ÜNÜVERSİTESİ
BİLİŞİM ENSTİTÜSÜ

TEMMUZ 2009
ANKARA

Gökmen ÇETİN tarafından hazırlanan EĞİTSEL BİR BİLGİSAYAR İÇİN DONANIM SİMÜLATÖRÜ VE SİMGESEL DİL BİRLEŞTİRİCİSİ 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 : Prof. Dr. Abdullah ÇAVUŞOĞLU

Üye : Prof. Dr. Doğan ÇALIKOĞLU

Üye : Yrd. Doç. Dr. Aslıhan TÜFEKÇİ

Tarih : 17/06/2009

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.

Gökmen ÇETİN

EĞİTSEL BİR BİLGİSAYAR İÇİN
DONANIM SİMÜLATÖRÜ VE SİMGESEL DİL BİRLEŞTİRİCİSİ
(Yüksek Lisans Tezi)

Gökmen ÇETİN

GAZİ ÜNÜVERSİTESİ
BİLİŞİM ENSTİTÜSÜ
TEMMUZ 2009

ÖZET

Bu çalışmada, BB3 (Bizim Bilgisayar 3) isimli eğitsel bir bilgisayar için bir donanım simülatörü ile bir birleştirici gerçekleştirilmiştir. Diğer bilgisayar simülatörlerine göre donanım simülatörünün farkı, amacın komut işleyişini simüle etmekten ibaret olmaması, simülasyonun en alttaki sayısal devre tasarımı düzeyinde olmasıdır. Böylece, simülatördeki deyimler ile bilgisayarın sayısal devreleri arasında karşılıklı ilişki kurulabilmektedir. Bu ilişki sayesinde de, sayısal devrelerde düşünülen her değişiklik, kolayca simülatöre yansıtılabilmekte ve denenebilmektedir. Simülatör ekranında BB3'ün hafızası ve kayıtlıkları ile birlikte, giriş ve çıkış birimlerini temsilen bir klavye ve bir ekran (BB3'e ait) yer almaktadır. İzlence güdümlü giriş/çıkış olduğu gibi, kesinti güdümlü giriş/çıkış da başarıyla gerçekleştirilmiş ve denenmiştir. Simülatörün esnekliğine paralel olarak, birleştirici de yeni komut simgeleri eklenebilir veya eskileriyle değiştirilebilir tarzda tasarımlanmıştır. Yazılım Windows işletim sistemi altında, Delphi 7 programlama dilinde hazırlanmıştır.

Bilim Kodu	**:**	**702.03.006**
Anahtar Kelimeler	**:**	**Donanım simülatörü, Donanım tasarımı, Kayıtlık aktarım dili, Yığıt, Birleştirici**
Sayfa Adedi	**:**	**95**
Tez Yöneticisi	**:**	**Prof. Dr. Doğan ÇALIKOĞLU**

HARDWARE SIMULATOR AND ASSEMBLER
FOR AN EDUCATIONAL COMPUTER
(M. Sc. Thesis)

Gökmen ÇETİN

GAZI UNIVERSITY
INFORMATICS INSTITUTE
July 2009

ABSTRACT

In this study, a hardware simulator and an assembler is realized for an educational computer called BB3 (Bizim Bilgisayar 3). A hardware simulator differs from the other simulators in that, simulating the operation of instructions is not the only purpose, but the simulation is performed at the lowest digital circuit design level, Thus, a correspondence can be established between the simulator statements and the computer's digital circuits. Due to this correspondence, every modification that is thought can be readily reflected to the simulator and tested. A keyboard and a screen (belonging to BB3) to represent the I/O units of BB3 is placed on the simulator's screen, together with its memory and registers. Just as the programed I/O, also interrupt driven I/O is realized and successfully tested. In parallel to the flexibility of the simulator the assembler is also designed in a way to enable adding new instruction symbols or replacing the old ones. The software is written in Delphi 7.0 programming language under the Windows operating system.

Science Code	**:**	**702.03.006**
Key Words	**:**	**Hardware simulator, Hardware design, Register transfer language, Stack, Assembler**
Page Number	**:**	**95**
Adviser	**:**	**Prof. Dr. Doğan ÇALIKOĞLU**

TEŞEKKÜR

Bu tez kapsamında geliştirilen simülatör ve birleştiricinin ilk prototip çalışmaları, on seneyi aşkın bir süredir danışmanım Prof. Dr. Doğan ÇALIKOĞLU tarafından "Bizim Bilgisayar" teması altında anlatılmakta olan donanım tasarımı konuları çerçevesinde başlatılmıştır. Bu süreç içerisinde bir taraftan, "Bizim Bilgisayar" olarak ortaya konan tasarım geliştirilmiş, diğer taraftan da öğrencilerin bu konuları deneysel bir tarzda daha kolay kavrayabilmeleri için yararlanılan simülatör ve birleştirici geliştirilmiştir. "Bizim Bilgisayar" tasarımında gelinen son model olan BB3'ün simülatör ve birleştiricisi, ilk bu tez çalışmasıyla meydana konmuş ve ilk defa, 2008-2009 öğretim yılı 2. yarıyılında Gazi Üniversitesi Endüstriyel Sanatlar Eğitim Fakültesi Bilgisayar Eğitimi Bölümünde okutulan "Donanım Tasarımı" dersinde laboratuar ortamında denenmiştir. BB3 simülatörünün ve birleştiricisinin hatalarının ayıklanmasında ve çeşitli yönlerden iyileştirilmesinde, söz konusu derse devam eden öğrencilerden alınan geri-besleme yararlı olmuştur. Ülkemizdeki eğitime bir katkı olması amaçlanan bu simülatör ve birleştiriciyi kullanmak isteyenlerin danışmanımdan izin almaları yeterlidir. Çalışmalarım boyunca bütün katkısı olanlara ve değerli yardım ve katkılarıyla beni yönlendiren danışmanım Prof. Dr. Doğan ÇALIKOĞLU'na 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 bir donanım simülatörü hazırlanmıştır. Bu simülatörü geliştirmek isteyecekler açısından yazılımın sistematik olmasına dikkat edilmiştir. Donanım Tasarımı dersini alan öğrenciler donanım simülatörünü inceleyebilecekler, kodları anlayacaklar, hatta kodlar üzerinde değişiklik yapabileceklerdir. Öğrencilerin bu simülatörün kodlarını anlamak ve simülatör üzerinde değişiklik yapabilmeleri için, simülatör öğrencilerin bildikleri bir programlama dili ile yazılmıştır. Bu simülatörün bir tamamlayıcı kolaylığı olarak, Bizim Bilgisayar Simgesel Dili (BBSD) ismi verilen simgesel dilin [2] kullanılabilmesi için bir Birleştirici de mevcuttur.

BB3 donanım işleyiş ilkelerinin öğretiminde kullanılmak üzere, Çalıkoğlu tarafından daha önce BB1 ve BB2 isimleri altında tanıtılan, öğretim amaçlı bilgisayar modellerinin daha gelişkin bir şeklidir [1].

Bizim Bilgisayar/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].

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.

Kayıtlık aktarım dilini Stepler "Microprogram Transformations" başlıklı makalesinde şöyle tanımlamaktadır: "Kayıtlık aktarım dili dijital sistemleri doğru ve kompakt bir şekilde anlatmak için kullanılır. Dijital sistemi oluşturan elemanların aralarındaki mantıksal ilişkiyi kayıtlık aktarım dili etkin bir şekilde gösterir" [3].

Kayıtlık aktarım dilinin uygulanışını Schorr "Computer-Aided Digital System Design and Analysis Using a Register Transfer Language" başlıklı makalesinde şöyle açıklamaktadır: "Dijital sistemlerde her bir kayıtlık bir karakter dizisi olarak tanımlanır ve n tane elemandan oluşmuş bir dizi olarak düşünülür. Örneğin R bir kayıtlık tanımlayıcısı ise R[i], i. flip-flop veya i. bit değeri gibi değerleri R kaydedicisinde tutar" [4].

Bizim Bilgisayar/2

Tasarlanan BB/1 bilgisayar modeli biraz daha geliştirilerek adına BB/2 denmiştir. BB/2'de iki adet yenilik yer almaktadır. İlk yenilik; Kesinti düzenindeki kesinti talebi kaynaklarının ferden izne bağlanmasıdır. Bu maksatla, her bayrak için, GBİ (Giriş Bayrağı İzni) ve ÇBİ (Çıkış Bayrağı İzni) şeklinde birer "izin" ikilisi tahsis edilmiştir. İkinci yenilik ise; Kayıtlık komutlarında kullanılan mikroişlemlerin zamanları yeniden düzenlenmiştir. Bu sayede birden fazla işlem aynı komutun içinde programlanabilmektedir [2].

Bizim Bilgisayar/3

Tasarlanan BB/2 bilgisayar modeli biraz daha geliştirilerek, BB/3 ismi verilen yeni bir model tasarlanmıştır. BB/3'ün BB/2'den temel farkı, BB/3'de yığıt yapısının kullanılmasıdır.

Intel X86 mimarisindeki yığıt yapısını Khan ve Anwar, "Design and Construction of a PC-Based Stack Machine Simulator for Undergraduate Computer Science & Engineering Courses" başlıklı makalesinde şöyle açıklamaktadır: "Intel X86 mimarisinde yığıt hafızanın üstünden itibaren aşağı doğru yerleşir. Yığıt işaretçisi yığıtın tepe değerini gösterir. Bu nedenle yığıta veri koyma işleminde, önce yığıt işaretçisi bir azaltılır, daha sonra veri yığıta yüklenir. Yığıttan veri çekilirken ise, yığıt işaretçisinin gösterdiği veri çekilir ve yığıt işaretçisi bir artırılır" [5].

BB/3'te altyordam yapısı değiştirilmiştir. Altyordam çağrısı olduğunda, altyordamdan dönüş adresi yığıta aktarılır. Aynı şekilde bir kesinti meydana geldiğinde, kesinti sonrası dönülecek adres yine yığıtta tutulur. Yığıt için 4K'lık ana hafızada belli bir alan ayrılmıştır. Yığıttan birikece veya birikeçten yığıta aktarılacak verinin adresini tutmak için 12 bitlik Yığıt İşaretçisi (Yİ) isimli yeni bir kayıtlık eklenmiştir. Yığıt işlemleri için BB/2'nin komutlarına 7 yeni komut daha eklenmiştir. Bu komutlar DÖN, YDY, YİA, YİE, Yİİ, YVA ve YVK olarak isimlendirilmişlerdir. Ayrıca BB/2 kayıtlıklarına Komut Kayıtlığı (KK) isimli 16 bitlik bir kayıtlık eklenmiştir [1].

Bizim Bilgisayar Simgesel Dili

Bizim Bilgisayar Simgesel Dili (BBSD) ÇALIKOĞLU 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 Donanım Simülatörü

Bu çalışma, daha önce benzeri Özlem Çakır tarafından yapılan donanım simülatörünün daha gelişkin bir şeklidir [6].

Bu çalışmada Delphi programlama dili kullanılarak BB/3 isimli eğitsel bir bilgisayar için bir donanım simülatörü ile bir birleştirici gerçekleştirilmiştir. Bu simülatör açık kaynak kodlu olup daha ileri geliştirmelere imkan tanımaktadır. Öğrencilerin Delphi programlama diline aşina olduklarından, simülatör yazılırken Delphi programlama dili seçilmiştir. Bu sayede Donanım Tasarımı dersini alan öğrenciler simülatör kodlarını inceleyip anlayabilirler ve hatta donanım üzerinde yapmak istedikleri değişiklikleri simülatöre uyarlayıp, simülatörü değiştirebilirler.

BB/3 simülatörünün 4K büyüklüğünde (her kelimesi 16 bitlik) ana hafızası vardır. Ana hafızayı temsil etmek için bir tablo kullanılmıştır. Simülatörün ana penceresinde bulunan bu tablo içerisinde 4K'lık hafıza adresleri ve bu hafıza adreslerinin verileri gösterilmektedir [3]. BBSD dili ile yazılmış kaynak kodun birleştirilmesiyle oluşturulan amaç kodların değerleri yine bu tablo içerisinde gösterilmektedir. Yığıtı daha kolay incelemek için birinci tablonun altında "FF0" dan "FFF" hafıza adresini gösteren ikinci bir tablo bulunmaktadır. Ayrıca simülatörün ana ekranında, giriş işlemlerinin yapılabilmesi için klavye, çıkış işlemlerinin sonuçlarının görülebilmesi için bir ekran bulunmaktadır. Simülatör ana ekranında BB/3 kayıtlıklarının, bayraklarının ve ikililerinin değerlerini gösterir yazı kutucukları bulunmaktadır. BB/3 izlencelerini zaman-zaman, devir-devir, komut-komut, serbest ve duraklı serbest olarak çalıştırabilmek için işlem düğmeleri bulunmaktadır. Bu düğmeler seçilerek izlenceler çalıştırıldığında Kİ, Yİ, HAK, HDK, KK, d, z, E, Bİ, GİRK, GİRB, GBİ, ÇIKK, ÇIKB, ÇBİ, KYV ve Ç kayıtlık ve bitlerinin alacağı değerler ve ana hafızada olan değişiklikler aynı pencere içersinde kullanıcıya gösterilir.

BB/3 simülatörü BB/1 simülatörünün tüm özelliklerine sahiptir. BB/1 simülatöründen farklı olarak giriş-çıkış ve yığıt işlemlerinin simülasyonu da yapılmaktadır. Simülatör BB/3 amaç kodları ile yazılmış izlenceleri çalıştırdığı gibi

BBSD dili ile yazılmış kaynak kodların amaç koduna dönüştürülüp çalıştırılabilmesi için gerekli bir *birleştiriciye* de sahiptir. BB/3 simülatörü ana ekranında yer alan bir buton sayesinde birleştirici penceresine ulaşılabilir.

Birleştirici

Öğrencilerin BBSD dilini kullanarak izlenceler yazabilmesi için donanım simülatörüne bir birleştirici eklenmiştir. Bu birleştirici sabit simgeler (BB/3 komutları), tanımlanmış simgeler (yaftalar), makrolar, sabit değerler, değer ifadeleri ve yardımcı komutlar kullanılarak, BBSD dilinde yazılmış izlencelerin amaç kodunu oluşturur ve simülatöre aktarır. Bu birleştirici sayesinde öğrenciler izlencenin kaynak kodlarını (simgesel kod) ve amaç kodlarını (makine kodu) irdeleyebilmektedirler ve böylece dil teorisi hakkında bilgi sahibi olmaktadırlar.

Çalışmanın Başlıca Özellikleri ve Önemi

Bu çalışmanın en önemli özeliği, oluşturulan donanım simülatörün bir tasarımdan öte öğrencilerin derslerinde kullanabildikleri bir simülatör olmasıdır. Bu birleştirici ve simülatör açık kaynak kodlu olup daha ileri geliştirmelere imkan tanımasından dolayı Donanım Tasarımı dersini alan öğrenciler, donanım simülatörünü inceleyebilecekler, kodları anlayacaklar, hatta kodlar üzerinde değişiklik yapabileceklerdir. Öğrencilerin bu simülatörün kodlarını anlamak ve simülatör üzerinde değişiklik yapabilmeleri için, simülatör öğrencilerin bildikleri bir programlama dili ile yazılmıştır.

Bu çalışmada yapılan BB/3 simülatörü BB/1 simülatörünün tüm özelliklerine sahiptir. BB/1 simülatöründen farklı olarak giriş-çıkış ve yığıt işlemlerinin simülasyonu da yapılmaktadır. Bu sayede öğrencilere giriş-çıkış işlemleri ve modern bilgisayarlarda kullanılan yığıt yapısı karmaşıklığa sebep olmadan basit bir şekilde öğretilebilir.

Bu simülatör, görsel bir programlama dili olan Delphi 7.0 ile oluşturulmuştur. BB/1 simülatörüne göre BB/3 simülatörünün kullanıcı arayüzü değiştirilmiştir. Bu

kullanıcı arayüzünde görsel programlama dillerinin kullanıcılara sağladığı işlem kolaylıkları sağlayacak özellikler yer almaktadır. Simülatörün arayüzünde Windows işletim sisteminde kullanılan nesneler ve menüler kullanılmıştır. Kullanıcılar bu nesne ve menülere aşina olduklarından dolayı simülatörü çok kolay kullanabilmektedirler. Bu program kaynak gösterilmek şartı ile eğitim amaçlı olarak herkes tarafından çoğaltılarak kullanılabilir.

Bölüm 2'de Bizim Bilgisayar/3 Bilgisayar Modeli ve Bizim Bilgisayar Simgesel Dilinin Temelleri, Bölüm 3'de BB/3 Simülatörünün ve Birleştiricinin Arayüzü, Bölüm 4'te Deneyler, Bölüm 5'te Sonuçlar, Öneriler ve İleride yapılabilecek Çalışmalar anlatılmaktadır. Ekler kısmında ise simülatörün kullanım kılavuzu ve simülatörün kaynak kodları vardır.

2. BİZİM BİLGİSAYAR/3 BİLGİSAYAR MODELİ VE BİZİM BİLGİSAYAR SİMGESEL DİLİNİN TEMELLERİ

Bizim Bilgisayar bilgisayar tasarım modelinin ilk versiyonu BB/1'dir. Daha sonra BB/1 geliştirilerek BB/2, BB/2 geliştirilerek BB/3 bilgisayar modelleri oluşturulmuştur.

2.1. Bizim Bilgisayar/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. Bilgisayar donanım tasarımı ve donanım işleyiş ilkelerinin daha etkili öğretilebilmesi için ÇALIKOĞLU tarafından bir bilgisayar tasarımı yapılmıştır [2]. Bu bölümde tasarlanan BB/1 isimli bilgisayar modelinin temel özellikleri anlatılmaktadır.

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].

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 kolaylıklar bulunması arzu edilen özelliklerdir.

Bir komutun ihtiva etmesi elzem olan iki bilgi vardır. Fiil ve nesne (veya nesneler). Bu iki bilgi, bir komutun içine ikilik kod halinde yerleştirilir [1].

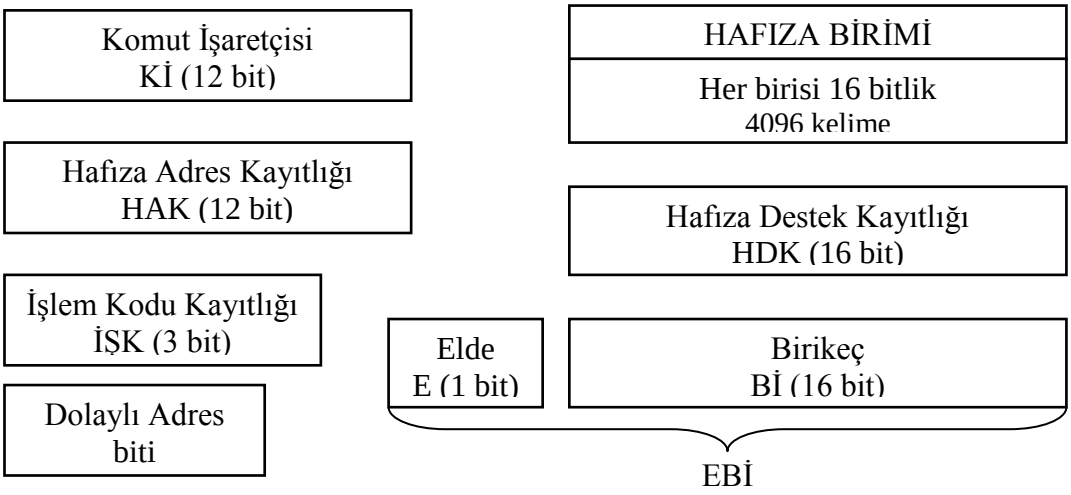
PDP/8 bilgisayarı 4K (4096) kelimelik bir ana hafızaya sahiptir. Ana hafızanın her her kelimesi 12 bit uzunluğundadır. Bütün aritmetik, mantık ve kaydırmaca işlemleri "birikeç" denen 12 bitlik bir kayıtlıkta yapılmaktadır [2].

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 değiştirmek ve çıkarış yapmak mümkün olmaktadır. Çarpım, kaydırarak toplayışla, bölüm, kaydırarak çıkartışla elde edilmektedir. İki kelimelik veya 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, "dolaylı adresleyiş" denen bir yöntem kullanılmaktadır. Dolaylı adresleyiş yönteminden, indisli adresleyişi gerçekleştirmekte de yararlanılmaktadır. İndisli adresleyiş için özel bir indis kayıtlığı bulunmayıp, hafıza yerlerinin her biri bu amaçla kullanılabilmektedir [2]. Bu yöntemde komut kelimesinin bir biti, adresleyiş tarzının doğrudan mı yoksa dolaylı mı olduğunu belirtmektedir. Doğrudan adresleyişte komutun adres kısmı, doğrudan hedef adrestir. Dolaylı adresleyişte ise komutun adres kısmı, hedef adresin bulunduğu yerin adresidir [1].

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].

2.1.1. BB/1'in temel kayıtlıkları

BB/1'in temel kayıtlıkları (Şekil 2.1) 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 2.1. BB/1 bilgisayarının ana hafızası ve temel kayıtlıkları.

2.1.2. BB/1'in komut kümesi

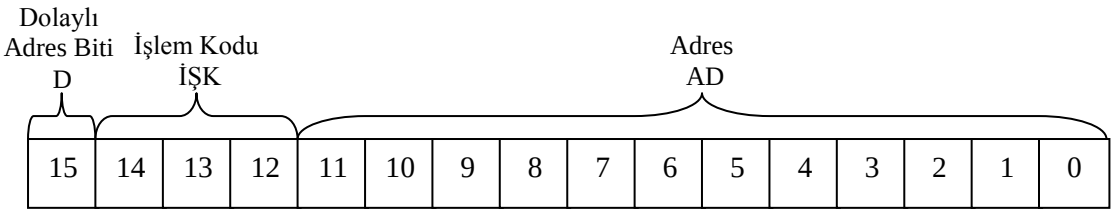
BB/1'in komutları aşağıdaki gibi sınıflandırılmıştır [1]:

A. Hafıza adresleyen komutlar
B. Hafıza adreslemeyen komutlar
   a) Kayıtlık komutları
   b) Giriş-çıkış komutları

BB/1'de; 6 adet hafıza adresleyen komut vardır (ASS, DÜS, SAK, SAP, TOP, VE ve YÜK). 12 adet kayıtlık komutu vardır (ART, BAS, BES, BSS, DUR, ESS, SAD, SLB, SLE, SOD, TMB ve TME). 4 adet giriş-çıkış komutu vardır (ÇBS, GBS, OKU ve YAZ).

2.1.3. Hafıza adresleyen komutlar

BB/1 bilgisayar modelinde, hafıza adresleyen komut kelimesi Şekil 2.2' de gösterilmiştir. 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 2.2. Bir hafıza adresleyen komut kelimesinin genel şekli.

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.

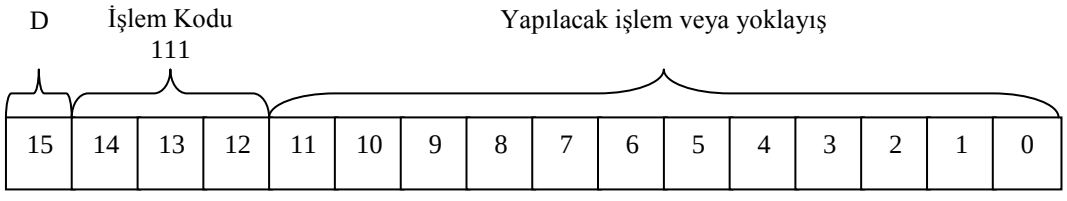
Hafıza adresleyen komutların işlem kodları, hatıraları (yani hatırlatan simgeleri), yaptıkları işlemler ve açıklamaları Çizelge 2.1'de görüldüğü gibi düzenlenmiştir [1].

Çizelge 2.1. Hafıza adresleyen komutlar

İŞ	Hatıra	Yaptığı İşlem	Açıklaması
000=0	VE	Bİ ← Bİ ^ (h)	Hedef hafıza yerindeki kelimeyi Bİ'ye 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

2.1.4. Hafıza adreslemeyen komutlar

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. Şekil 2.3'de görüldüğü gibi D bitinin "0" olması durumunda bu komut bir kayıtlık komutu olarak değerlendirilir, "1" olması durumunda ise komut giriş-çıkış komutu olarak değerlendirilir [1].



Şekil 2.3. Bir hafıza adreslemeyen komut kelimesinin genel şekli

2.1.5. Kayıtlık komutları

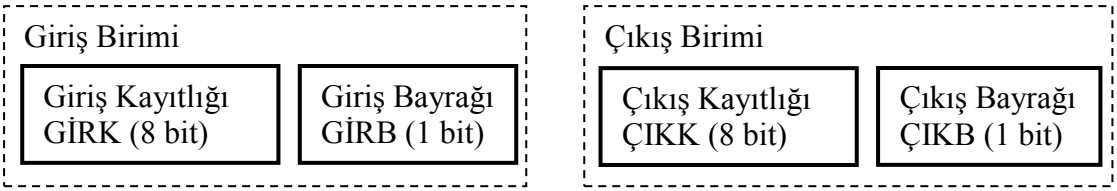
Kayıtlık komutlarında en soldaki dört bit sabitlenmiştir (0111). Geriye kalan oniki bitin farklı birleşimlerine farklı anlamlar vererek 4096 adete kadar farklı komut tanımlamak mümkündür. Ancak büyük kod çözücülere ve aşırı karmaşıklığa gerek bırakmamak için bu oniki bitin her birine diğerlerinden bağımsız 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	Hatıra	Yaptığı İşlem	Açıklaması
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.

2.1.6. Giriş-çıkış komutları

BB/1'de giriş birimi klavye, çıkış birimi de yazıcı olarak kabul edilmektedir. Bu birimlerle veri alış verişi, yapmak için Şekil 2.4'te görülen kayıtlıklar kullanılmaktadır [2].



Şekil 2.4. 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'i "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

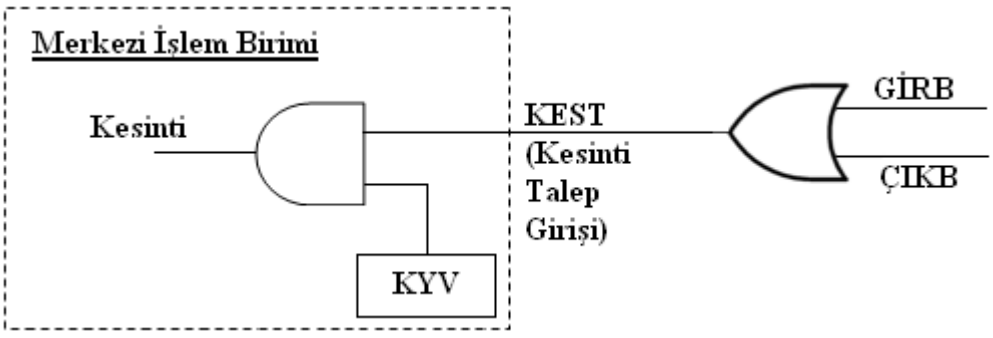
İlgili bit	Komut Kodu	Hatıra	Yaptığı İşlem	Açıklaması
11	F800	OKU	Bİ(7-0) ← GİRK, GİRB← 0	Bİ'ye bir bayt oku
10	F400	YAZ	ÇIKK ← Bİ(7-0), ÇIKB← 0	Bİ'den bir bayt yaz
9	F200	GBS	GİRB=1 ise Kİ ←Kİ+1	GİRB=1 ise sek
8	F100	ÇBS	ÇIKB=1 ise Kİ ←Kİ+1	ÇIKB=1 ise sek

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. İkinci 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 2.5'de 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 2.5. KEST 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. Ayrıca Çizelge 2.4'ün her satırında; ilgili bitin hangisi olduğu, o bitin bir olmasına tekabül eden komutun kodu, kodun hatırası, komutun yaptığı işlem ve komutun açıklaması bulunmaktadır [1].

Çizelge 2.4. Kesintiye izin komutları

İlgili bit	Komut Kodu	Hatıra	Yaptığı İşlem	Açıklaması
7	F080	KEV	KYV← 1	Kesintiye imkan var.
6	F040	KEY	KYV← 0	Kesintiye imkan yok

2.1.7. 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
```

2.1.8. 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, d_0, d_1, d_2, d_3 olarak belirtilmektedir [2]. Çizelge 2.5'de 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

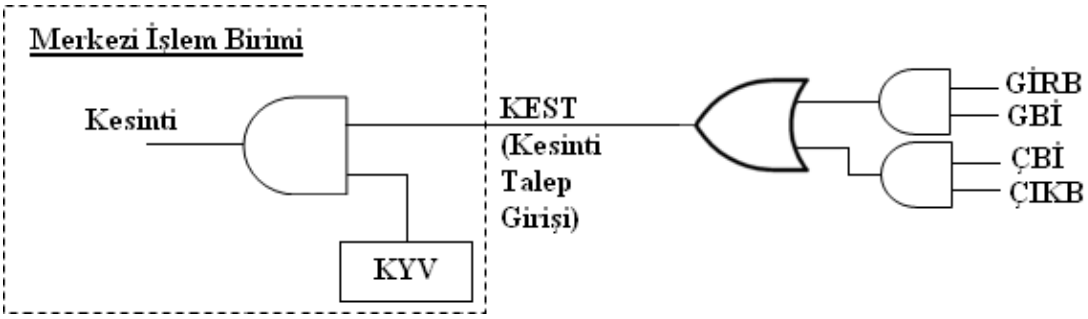
Kayıtlık devrelerini yazıyla tarif etmek için kayıtlık aktarım dili kullanılır. BB/1'in işleyişinin mantıksal tasarımında da kayıtlık aktarım dili kullanılmıştır. Bu dildeki deyimler şu genel biçimde olmaktadırlar [2]:
(Mantık ifadesi)**:** (mikroişlem), (mikroişlem),..., (mikroişlem)

Örneğin, d_0z_1**:** HDK $\rightarrow$ H, Kİ $\rightarrow$ Kİ + 1
Bu deyimde, d_0z_1 şartı sağlandığı anda,
HDK $\rightarrow$ H ve Kİ $\rightarrow$ Kİ + 1 mikroişlemlerinin 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.

2.2. Bizim Bilgisayar/2

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

1. Kesinti düzenindeki kesinti talebi kaynakları ayrı ayrı izine bağlanmıştır. Bu maksatla her bayrak için, GBİ (Giriş Bayrağı İzni) ve ÇBİ (Çıkış Bayrağı İzni) şeklinde birer "izin" ikilisi getirilmiştir ve KEST girişi Şekil 2.6'da görüldüğü gibi KEST=GBİ•GİRB+ÇBİ•ÇIKB mantığıyla yeniden düzenlenmiştir. Bu yeni düzenlemeyle bayraklardan istenmeyenlerin kesinti talebinde bulunmaları önlenebilmektedir [2].



Şekil 2.6. BB/2'de KEST 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	Hatıra	Yaptığı İşlem	Açıklaması
5	F020	GBV	GBİ ← 1	Giriş Bayrağı İzni Var
4	F010	GBY	GBİ ← 0	Giriş Bayrağı İzni Yok
3	F008	ÇBV	ÇBİ ← 1	Çıkış Bayrağı İzni Var
2	F004	ÇBY	ÇBY← 0	Çıkış Bayrağı İzni Yok

2. Kayıtlık komutlarında kullanılan mikroişlemlerin zamanları yeniden düzenlenmiştir. 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
SLB	z_0
SLE	z_0
TMB	z_1
TME	z_1
SAD	z_3
SOD	z_3
ART	z_2
BAS	z_0
BES	z_0
BSS	z_0
ESS	z_0
DUR	z_3

2.3. Bizim Bilgisayar/3

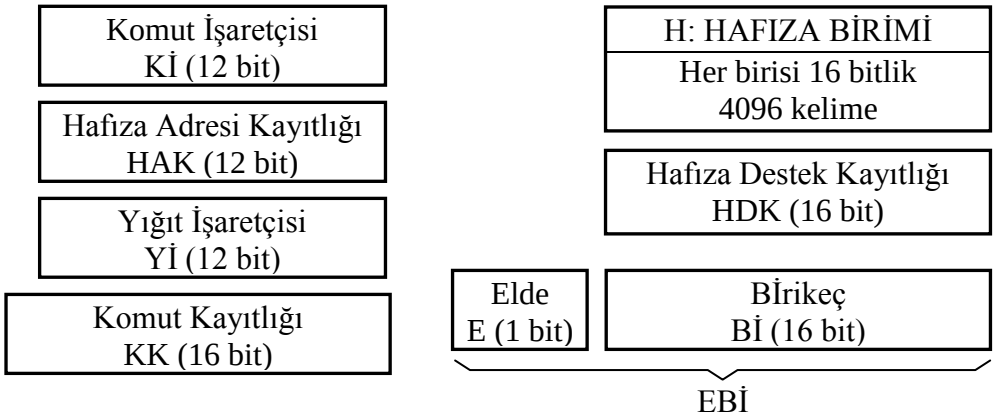
BB/2 bilgisayarı Çalıkoğlu tarafından geliştirilerek yığıt yapısı ilave edilmiş ve altyordam çağrıları için yığıt kullanılan bir model elde edilmiştir. Bu yeni model BB/3 olarak adlandırılmaktadır [1].

Yığıt için hafızada belirli bir alan ayrılmıştır. Yığıta bir sonra konulacak olan verinin, yığıtın hangi adresine konulacağını tutmak için BB/1 in mevcut kayıtlıklarına Yığıt işaretçisi (Yİ) adı verilen 12 bitlik yeni bir kayıtlık eklenmiştir. Yİ kayıtlığı, yığıtın tepesinin adresini izlemekte kullanılır. Yığıta veri aktarma ve yığıttan veri alma işlemleri bu kayıtlığın gösterdiği hafıza adresine göre yapılır [1].

BB/3'ün BB/2'den temel olarak beş adet farkı vardır:

1. BB/l 'deki DUR komutu gereksiz görüldüğü için kaldırılmıştır. BB/1 'in komutlarına yığıt komutları olarak DÖN, YVA, YVK, YİA, YİE, Yİİ ve YDY isimli komutlar eklenmiştir. DÖN komutu, DÜS komutuyla altyordam çağrısı olduğunda, altyordamdan program akışında kalınan yere geri dönebilmek için kullanılır. YVA komutu, yığıttan birikeçe veri alınmasını sağlar. YVK komutu, birikeçteki veriyi yığıta aktarmayı sağlar. YİA komutu Yİ'nin değerini bir artırmak için, YİE komutu Yİ'nin değerini bir azaltmak için, Yİİ komutu yığıtın işaret ettiği tepe değeri bir arttırmak için, YDY ise yığıttan birikece dolaylı veri alınmasını sağlar [1].

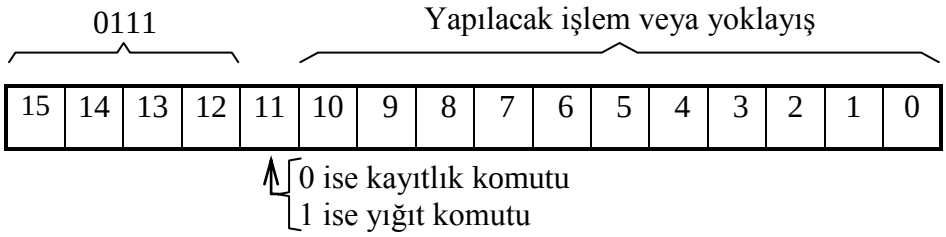
2. BB/2 kayıtlıklarına KK(Komut Kayıtlığı) isimli 16 bitlik bir kayıtlık eklenmiştir. Bu sayede İŞK ve D kayıtlıklarına ihtiyaç ortadan kalkmıştır [1].



Şekil 2.7. BB/3 bilgisayarının ana hafızası ve temel kayıtlıkları.

3. 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. Yedi adet yığıt komutu

tanımlanmıştır. Bunlar DÖN, YVA, YVK, YİA ve YİE, Yİİ ve YDY'dir. Yığıt komutları için bit1, bit2 ve bit3 bitleri tahsis edilmiştir [1].



Şekil 2.8. BB/3 bilgisayarının bir yığıt veya kayıtlık komut kelimesi

Böylece bir kayıtlık veya yığıt komutu kelimesinde, yapılacak işlem veya yoklayışı ifade edebilmek için, bit10 -dan bit0 -a kadar onbir bitlik bir kısım kalmaktadır. Söz konusu onbir bitin her birine Çizelge 2.8 de görüldüğü şekilde birer işlem tayin edilip Temel Kayıtlık Komutları oluşturulmuştur. Bu çizelgede cetvelin her satırında ilgili bitin hangisi olduğu, o bitin bir olmasına tekabül eden komut kodu, hatırası, yaptığı işlem, açıklanışı ve o işleme ait olan bir "zaman sırası" görülmektedir [1].

Çizelge 2.8. BB/3'de Temel Kayıtlık Komutları ve işlem zamanları

İlgili bit	Komut Kodu	Hatıra	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

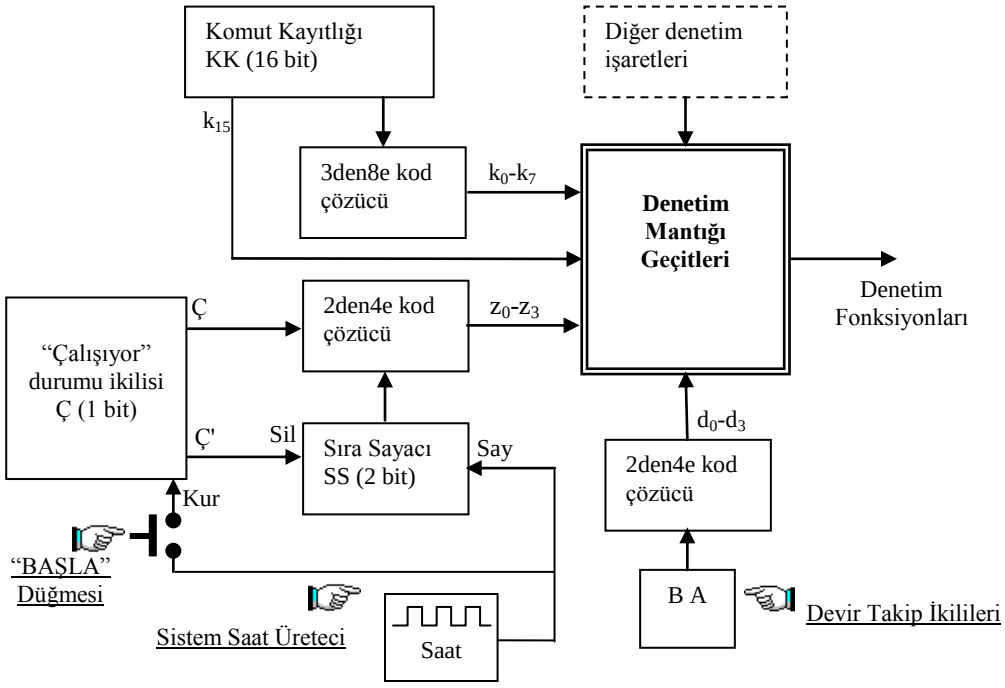
Şekil 2.8'de kayıtlık komut kelimesinin genel şekli görünmektedir. Böylece bir kayıtlık veya yığıt komutu kelimesinde en soldaki 5 bit sırasıyla 01111 olduğunda

bunun bir yığıt komutu olduğunu anlıyoruz. Geri kalan onbir bitin her birine Çizelge 2.9 da görüldüğü şekilde birer işlem tayin edilip Yığıt Komutları oluşturulmuştur. Bu çizelgenin her satırında ilgili bitin hangisi olduğu, o bitin bir olmasına tekabül eden komut kodu, hatırası, yaptığı işlem ve açıklanışı görülmektedir [1].

Çizelge 2.9. BB/3'de Temel Kayıtlık Komutları ve işlem zamanları

İlgili bit	Komut Kodu	Hatıra	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, (Yİ)←Bİ	Yığıta veri koy(gönder)
3	7808	YVA	Bİ←(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.

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



Şekil 2.9. BB/3 bilgisayar modeli denetim mantığının ilkesel görünümü.

4. BB/3‘te altyordama sapış ve altyordamdan dönüş mantığı değiştirilmiştir. BB/1’ de altyordamdan dönüş adresi, altyordamın başlangıç adresinde tutulmaktaydı. Böylece, altyordamdan ana programda kalınan yere geri dönebilmek için D SAP komutuyla bu adrese dolaylı olarak sapılırdı. BB/3’te alt yordamdan dönüş adresi yığıtta tutulur. Bundan dolayı, altyordamın başlangıç adresinin bu iş için kullanılmasına gerek kalmamıştır. Ayrıca, BB/l‘de altyordamdan dönüş için kullanılan D SAP komutu yerine, yığıtta tutulan dönüş adresini alıp bu adresten devam etmeyi DÖN komut sağlar. Ana izlencede DÖN komutu kullanıldığında ise, BB/1 deki DUR komutunun yaptığı işlem gerçekleşir. Şekil 2.10 da BB/3’de kullanılan altyordam yapısı gösterilmiştir [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 ÇAĞRISI YAPILAN YER.
YER 200
ÇAĞRI: DÜS ALTYOR
DEVAM: --- / “DÜS” KOMUTUNUN ALTINDAKİ KOMUT
```

Şekil 2.10. BB/3 bilgisayar modelinde altyordam yapısı

5. BB/3’de kesinti düzeni değiştirilmiştir. BB/3‘te kesinti olduğunda, kesintiden sonraki dönüş adresi yığıta aktarılır [1].

2.3.1. 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.			**dolaylı**	d_1z_0:	HAK← KK(AD)
				d_1z_1:	HDK← H
				d_1z_3:	B← 1, A← 0
getir	d_0z_0:	HAK← Kİ	**kesinti**	d_3z_0:	HDK(AD)←Kİ, Kİ← 0, Yİ←Yİ-1
	d_0z_1:	HDK← H, Kİ← Kİ + 1		d_3z_1:	HAK← Yİ
	d_0z_2:	KK← HDK, HAK← Yİ		d_3z_2:	H← HDK, KYV← 0
	$(v_{15}k_7')d_0z_3$:	A← 1 {BA=01, d_1=1 olur.}		d_3z_3:	B← 0, A← 0 {BA=00, d_0=1 olur.}
	$(v_{15}k_7')'d_0z_3$:	B← 1 {BA=10, d_2=1 olur.}			

İfa

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← Bİ
	$k_2d_2z_2$:	Bİ← Bİ ∧ HDK		$k_3d_2z_2$:	H← HDK
SAP	$k_4d_2z_0$:	Kİ← HDK(AD)	**DÜS**	$k_5d_2z_0$:	HDK(AD)←Kİ, Kİ← HDK(AD), Yİ←Yİ- 1
ASS	$k_6d_2z_0$:	HAK← HDK(AD)		$k_5d_2z_1$:	HAK← Yİ
	$k_6d_2z_1$:	HDK← H		$k_5d_2z_2$:	H← HDK
	$k_6d_2z_2$:	HDK← HDK + 1			
	$k_6d_2z_3$:	H← HDK, Eğer (HDK=0) ise (Kİ← Kİ + 1)			

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.			**SLE**	v_5r_0:	E← 0
BAS	$v_{10}r_0$:	Eğer (Bİ(15)=0) ise (Kİ← Kİ + 1)	**TMB**	v_4r_1:	Bİ← (Bİ)'
BES	v_9r_0:	Eğer (Bİ(15)=1) ise (Kİ← Kİ + 1)	**TME**	v_3r_1:	E← E'
BSS	v_8r_0:	Eğer (Bİ=0) ise (Kİ← Kİ + 1)	**ART**	v_2r_2:	EBİ← Bİ + 1
ESS	v_7r_0:	Eğer (E=0) ise (Kİ← Kİ + 1)	**SAD**	v_1r_3:	Sağd EBİ
SLB	v_6r_0:	Bİ← 0	**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.			**YDY**	v_2t_0:	HDK← H
YİE	v_6t_0:	Yİ←Yİ-1		v_2t_1:	HAK←HDK(AD), Bİ←0
YİA	v_5t_2:	Yİ←Yİ+1		v_2t_2:	HDK← H, Bİ← (Bİ)'
				v_2t_3:	Bİ← Bİ ∧ HDK
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
YVA	v_3t_0:	Bİ← 0	**DÖN**	v_0t_0:	HDK← H
	v_3t_1:	HDK← H, Bİ← (Bİ)'		v_0t_1:	Kİ← HDK(AD)
	v_3t_2:	Bİ← Bİ ∧ HDK, Yİ←Yİ+1		v_0t_2:	Yİ←Yİ+1
				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:	GİRB = 1 ise Kİ← Kİ+1	**GBY**	v_4s:	GBİ← 0
ÇBS	v_8s:	ÇIKB = 1 ise Kİ← Kİ+1	**ÇBV**	v_3s:	ÇBİ← 1
KEV	v_7s:	KYV← 1	**ÇBY**	v_2s:	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}

2.3.2. Bizim bilgisayar simgesel dili (BBSD)

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 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].

Simgesel programlar, makine dili programlar gibi alt alta yazılan simgesel komutlardan oluşur. İzlencelerin makine dili şeklinde olan kodlarına "Amaç kodu", simgesel komutlar şeklinde olan kodlarına "Kaynak kodu" demekteyiz. Simgesel komutlar, makine dili karşılığı olanlar ve olmayanlar diye iki cins olur. Makine dili olmayan simgesel komutlara *yardımcı komutlar* denir. Yardımcı komutlar birleştiriciye hitap ederler.

Bir programın farklı yerlerinde tekrarlanan bölümleri tekrar tekrar yazmak yerine işletim siteminin makro işleme özelliği kullanılır. Makro işleminde yapılacak işlemler makro tanımlama bölümünde kodlanır ve bir isim altında kaydedilir. Programın çeşitli bölümlerinde bu isim çağırılarak makro tanımlamada yazılan kodlar çalıştırılır [7].

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

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

2. *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].

3. *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].

4. *Makro*: Herhangi bir satırın solunda bir harfle başlayan, iki defa eşittir (==) işareti ile biten ve arada boşluk bulunmayan bir harf rakam dizgisi makro olarak tanımlanmıştır. Makronun muhtevası ise iki eşittir işaretinin sağında bulunan komutların muhtevalarının VEYA'lanmasıyla oluşan değere eşittir.

5. *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].

BBSD'nin tanımı, bir sayfaya sığacak kısalıkta olmakla birlikte, bir simgesel dilde olması gereken tüm özellikler BBSD'nde mevcuttur.

BBSD'nin tanımı [2]:

1. Her satırda en çok bir hafıza yeri muhtevasının (komut veya veri) ifadesi olabilir.

2. Makine komutlarının hafızada denk geleceği yerin tayini için, sıradaki makine komutlarının hangi yerden itibaren dizilmesi istendiği, bir YER yardımcı komutuyla belirtilir. Ör. YER 200

3. Makine komutlarının sayısal kodları yerine simgeleri yazılabilir.

4. 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.

5. 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.

6. 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. 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.

   Örnek: K1: ART

   SAP K1

7. Simgesel programa açıklayıcı ifadeler ilave edebilmek için "/" işareti kullanılır. Bu işaretin sağında kalan şekilcikler makine diline çevrilişte dikkate alınmaz.

8. BBSD'de izlence yazarken birden çok komuttan oluşan bir satır, izlencede birden çok tekrarlanıyorsa bu satıra tekabül eden bir makro oluşturulabilir. Makroya bir isim verilir daha sonra yan yana iki defa eşittir (==) işareti konulur ve makroya tekabül eden komutlar yazılır ve makronun muhtevası oluşur. Makro izlencenin başka yerlerinde çağrıldığında çağrılan yere makronun muhtevası atanır.
   Örneğin; TAMLA == TMB ART olarak tanımlanan TAMLA makrosunun muhtevası "7014" olur. İzlencenin başka bir yerinde TAMLA denildiğinde oraya "7014" muhtevası atanır.

9. Hafıza adresleyen komutlarda hatıradan önce bulunan bir D harfi, indisli adresleyişin olduğunu belirtir.

10. Simgesel programın sonunu belirtmek için SON yardımcı komutu kullanılır.

Geleneksel derleyiciler 3 ana bölümden oluşur. Öncelikle giriş dosyası bir string diziye aktarılır (Lexical analyzer). Sonra bu dizideki elemanlar anlamlı parçalara (token) bölünür (Parser). Son olarak da derleyici bu parçalara uygun makine kodlarını üretir (Code Production) [8].

Derleyiciler çalışırken kodların üzerinden bir, iki veya daha fazla geçiş prensibine göre çalışabilirler. Değişkenlerin tanımlanmadan önce çağrılmasına olanak vermek için bir geçişli derleyiciler kullanılamaz. İki geçişli derleyicilerde birinci geçişte sadece değişkenler tanımlanır. İkinci geçişte ise adresler ve kodlar üretilir [7].

BB/3 simülatöründe üç geçişli bir derleyici hazırlanmıştır. İlk safhada makrolar tanımlanır. İkinci safhada programın üzerinden gidilerek yaftaların adres değerleri belirlenir ve bir simge cetveli oluşturulur. Üçüncü safhada programın üzerinden tekrar gidilerek, bu defa simge cetvelindeki değerlere göre komut kelimelerinin sayısal karşılıkları tespit edilir.

3. BB/3 SİMÜLATÖRÜ VE BİRLEŞTİRİCİ

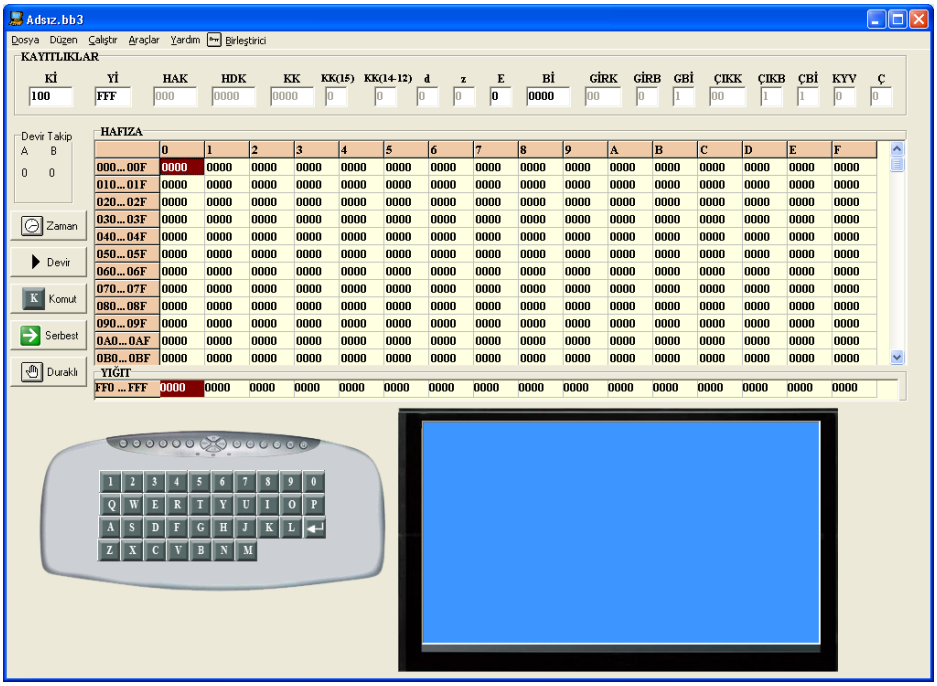
Delphi 7.0 ile oluşturulmuş olan BB/3 donanım simülatörü 4 pencereden oluşmaktadır. Bu pencereler; BB/3 simülatörü, Birleştirici, Sabit simge cetveli ve Sayı çeviricidir.

3.1. BB/3 Simülatörü

BB/3'ün işleyişinin mantıksal tasarımını irdelemek için, "donanım tasarım dili"ndeki anlatımı Delphi 7.0 diline çevrilerek bir donanım simülatörü oluşturulmuştur. Bu simülatörü geliştirmek isteyecekler açısından yazılımın sistematik olmasına dikkat edilmiştir. Bu sayede Donanım Tasarımı dersini alan öğrenciler simülatör kodlarını inceleyip anlayabilirler ve hatta donanım üzerinde yapmak istedikleri değişiklikleri simülatöre uyarlayıp, simülatörü değiştirebilirler.

BB/3 simülatörünün 4K büyüklüğünde (her kelimesi 16 bitlik) ana hafızası vardır. BB/3 simülatörünün ana hafızası; Const Azami_adr=4095; HAFIZA: Array "[0..Azami_adr] of string[4]" şeklinde tanımlanmıştır. Ana hafızayı temsil etmek için bir tablo kullanılmıştır. Simülatörün ana penceresinde bulunan bu tablo içerisinde 4K'lık hafıza adresleri ve bu hafıza adreslerinin verileri gösterilmektedir. BBSD dili ile yazılmış kaynak kodun birleştirilmesiyle oluşturulan amaç kodların değerleri yine bu tablo içerisinde gösterilmektedir. Yığıtı daha kolay incelemek için birinci tablonun altında "FF0" dan "FFF" hafıza adresini gösteren ikinci bir tablo bulunmaktadır. Ayrıca simülatörün ana ekranında, giriş işlemlerinin yapılabilmesi için klavye, çıkış işlemlerinin sonuçlarının görülebilmesi için bir ekran bulunmaktadır. BB/3 donanım simülatöründe 16 bitlik kayıtlıklar word, 8 bitlik kayıtlıklar byte, 12 bitlik kayıtlıklar onikibit, bayraklar ise bit olarak tanımlanmıştır. Bir kayıtlığın denetim girişlerinden birisinin etkin olmasının sonucu aynı anda yani "0" sürede çıkıştan almak mümkün değildir. Bu nedenle her kayıtlığın giriş ve çıkışını temsil etmek için, kayıtlıkların girişi ve çıkışı ayrı ayrı tanımlanmıştır.
Örneğin; BI_G (Birikeç kayıtlığının girişi), BI_C (Birikeç kayıtlığının çıkışı).

Simülatör ana ekranında BB/3 kayıtlıklarının, bayraklarının ve ikililerinin değerlerini gösterir yazı kutucukları bulunmaktadır. Bu yazı kutucuklarından Kİ, Yİ, E ve Bİ değerleri kullanıcı tarafından istenildiğinde değiştirilebilir. BB/3 izlencelerini zaman-zaman, devir-devir, komut-komut, serbest veya duraklı serbest olarak çalıştırabilmek için işlem düğmeleri bulunmaktadır. Bu sayede seçime bağlı olarak, her "zaman adımı", devir veya komut bitiminde Kİ, Yİ, HAK, HDK, KK, d, z, E, Bİ, GİRK, GİRB, GBİ, ÇIKK, ÇIKB, ÇBİ, KYV ve Ç kayıtlık ve bitlerinin alacağı değerler, ana hafızada ve çıkış ekranında olan değişiklikler aynı pencere içersinde, kullanıcıya gösterilir. Ayrıca kullanıcının daha kolay takip edebilmesi için, Kİ ve Yİ'nin işaret ettiği hafıza hücreleri renklendirilmiştir. Kİ'nin işaret ettiği hafıza hücresi sarı renkle, Yİ'nin işaret ettiği hafıza hücresi ise açık mavi renkle gösterilmiştir. Hücreleri renkli gösterebilmek için Delphi 7.0 programına Coolcontrol paketi yüklenmiştir. Bu paketin nasıl yüklendiği ekler kısmında açıklanmaktadır. Resim 3.1'de BB/3 donanım simülatörünün ana ekranı gösterilmektedir. B/3 simülatörü BB/1 simülatörünün tüm özelliklerine sahiptir. BB/1 simülatöründen farklı olarak giriş-çıkış ve yığıt işlemlerinin simülasyonu da yapılmaktadır.



Resim 3.1. BB/3 donanım simülatörünün ana penceresi.

Simülatör çalıştığında kayıtlıklara, bayraklara ve bitlere BB/3 bilgisayar modeline göre, Resim 3.1'de görünen ilk değerleri aktarılır. Yığıt işaretçisi ilk başta yığıtın en dibi olan "FFF" değerini gösterir. Herhangi bir veri girişi olmadığından ana hafızanın tüm kelimelerine "0000" değeri atanır.

Herhangi bir BB/3 izlencesinin yürütülebilmesi için öncelikle izlencenin makine kodlarının (Amaç kodlarının) BB/3 simülatörü ana hafızasına yüklü olması gerekir. Bunun için makine kodları BB/3 simülatörü ana hafıza kelimelerine girilebilir veya daha önce amaç kodları şeklinde kaydedilmiş olan izlenceler "Aç" penceresinden seçilerek yüklenebilir. Üçüncü bir yöntem olarak birleştirici kullanılarak simgesel kodlar kullanılarak yazılmış izlencelerin amaç kodları BB/3 simülatörü ana hafızasına aktarılabilir. BB/3 simülatörü ana hafızasına yüklenen izlence kodları "zaman_zaman_calistir" isimli bir altyordam vasıtasıyla, BB/3 izlencesinin komut işaretçisinde belirtilen adresten itibaren yürütülmesine başlanmaktadır. BB/3 simülatörü işlemcisinin çalışır durumda olup olmadığını çalış biti (C) ile belirliyoruz. C=1 olduğu sürece "zaman_zaman_calistir" altyordamı çalışmaya devam edebilmektedir. C=0 olduğunda BB/3 işlemcisi duracağından, ana hafızasında bulunan izlencelerin yürütülmesi de durur.

"zaman_zaman_calistir" altyordamı çalıştığında ilk başta Kayıtlıkların çıkış değerleri, bayrakların, bitlerin ve devir takip ikilisinin (A, B) değerleri aktarılır. Daha sonra kod çözücüsü değerleri aktarılır:

```
For i:=0 To 7 do K[i]:=0;
K[(KK and $7000) SHR 12]:=1;              {İşlem Kodu Kod Çözücü}
For i:=15 downto 0 do  V[i]:=(KK SHR i) and $0001; {KK nin  bit değerleri}

For i:=0 To 3 do Begin D[i]:=0;Z[i]:=0; End;
   D[2*B+A]:=1;      {BA Kod Çözücü}
   Z[SS]:=1;         {SS Kod Çözücü}
```

Kod çözücülerin önceki değerleri “0”landıktan sonra, İşlem Kodu kod çözücüsünün çıkış değeri “1”lenir. K[7]=1 olursa ifa edilecek komutun kayıtlık, giriş çıkış veya yığıt komutu olduğunu belirtir. K[7]’den başka bir çıkış “1”lenirse ifa edilecek komutun hafıza adresleyen komut olduğunu belirtir. Daha sonra “B” ve “A” devir takip ikilisine göre Devir belirlenir. D[0]:=1 değeri getir devrini, D[1]:=1 değeri dolaylı devri, D[2]:=1 değeri ifa devrini, D[3]:=1 değeri ise kesinti devrini belirtir. Devir belirlendikten sonra SS kod çözücüsü (Sıra sayacı) değerine göre zaman belirlenir. Daha sonra KK’nın uygun bitleri “1”lenir. Kod çözücü çıkışlara uygun devir ve zamandaki kayıtlık aktarım dili satırları taranır ve şartları geçerli olan mikroişlemler gerçekleştirilir. Komut işaretçisi olan Kİ’nin bir zaman adımı içinde birden fazla ilerletilmesi hatasını engellemek için KI_SAY biti kullanılır. Herhangi bir mikroişlem sonucu KI_SAY bit değeri “1” olmuş ise Kİ ilerletilir. Kayıtlıkların girişine gelen değerler çıkışa aktarılır. Kayıtlık, bayrak, bit ve ikili değerleri ekranda sergilenir. Sıradaki zamana geçilir. BB/3 simülatörünün ana hafıza kelimelerinin muhtevaları güncellenir.

“Zaman_zaman_calistir” altyordamının genel mantığı aşağıdaki gibidir:

1. Bit, bayrak, ikililer ve kayıtlık çıkışlarının değerlerini aktar.
2. Simüle edilecek devire ve zamana göre kod çözücü çıkış değerlerini belirle.
3. Kayıtlık aktarım dili satırlarını tara, şartları geçerli olan mikroişlemleri gerçekleştir.
4. Kayıtlık girişlerine gelen değerleri, kayıtlık çıkışlarına aktar.
5. Bit, bayrak, ikililer, kayıtlık değerleri, BB/3 simülatörü ana hafızası ve eğer varsa çıkış aygıtındaki değerleri sergile.
6. Sıradaki zamana geç.
7. İşleyiş devam edecek ise 2 no.lu satıra git, yoksa çık.

Yığıt işaretçisi, yığıtın dibi olan “FFF” değerini gösteriyorken “DÖN” komutu kullanıldığında yığıt işaretçisinin değeri “1000” olur. Dön komutunun son zamanında Yİ’nin bit11 değeri kontrol edilir. Eğer Yİ’nin bit11 = 0 ise çalış biti (C=0) sıfır yapılarak izlencenin bittiğini gösterir.

BB/3 simülatörü giriş birimi olan klavyeden bir tuşa basıl
dığında, tuşun şekilcik kodu Giriş kayıtlığına (GİRK) aktarılır. Giriş donanımı, GİRK'ya yeni bir şekilcik kod değeri geldiğini belirtmek için GİRB'i "1" yapar. OKU komut ile GİRK'daki şekilcik kodu birikece aktarılabilir. OKU komutu, GİRK'daki şekilcik kodunu birikece aktarır ve GİRK'da taze kod olmadığını göstermek için GİRB'yi "0" yapar. Eğer yazıcı müsait ise (ÇIKB=1) YAZ komutu ile birikece alınan şekilcik kodunu ÇIKK'ya aktarır. YAZ komutu aynı zamanda yazıcının meşgul durumuna geçtiğini belirtmek için ÇIKB'yi "0" yapar. Yazıcı yazma işlemini bitirdikten sonra tekrar müsait duruma geçtiğini göstermek için ÇIKB'yi "1" yapar. BB/3 simülatörü gerçeğe uygun bir simülatördür. Yazıcının donanım gecikmesini devreye sokmak için YAZ komutu ifa edilirken, Print_suresi isimli bir değişkenin değeri "100" yapılır. "Print_sureci" isimli bir altyordam yardımıyla, Print_süresi değişkeninin değeri sıfır oluncaya kadar her zamanın sonunda birer azaltılır. Değişkenin değeri sıfır olduğunda, kodu ÇIKK'da olan şekilcik, BB/3 simülatör penceresindeki çıkış birimine yazılır.

3.1.1. Dosya menüsü

Dosya menüsünde Aç, Kaydet, Farklı Kaydet ve Çıkış seçenekleri vardır.

Aç komutu seçildiğinde "AÇ" penceresi açılır. Bu pencere; daha önce kayıtlı olan, makine kodları ile yazılmış BB/3 (bb3 uzantılı) izlencelerini, BB/3 simülatöründeki ana hafızaya aktarmaya sağlar.

Kaydet komutu seçildiğinde "KAYDET" penceresi açılır. Bu pencere, BB/3 simülatöründeki ana hafızada bulunan makine dili kodlarını bir dosyaya kaydetmek için kullanılır. Dosyanın uzantısı "bb3" olur. Eğer kaydedilecek dosya, daha önce kayıtlı ise dosyanın içeriği güncellenir. Dosya daha önce kayıtlı değilse, ilk defa kaydediliyorsa "FARKLI KAYDET" penceresi açılır.

Farklı Kaydet komutu seçildiğinde "FARKLI KAYDET" penceresi açılır. Bu pencere, BB/3 simülatöründeki ana hafızada bulunan makine dili kodlarını farklı bir

yerdeki farklı bir dosyaya kaydetmek için kullanılır. Dosyanın uzantısı “bb3” olur. Eğer kaydedilecek dosya, daha önce kayıtlı ise dosyanın içeriği güncellenir.

Çıkış komutu seçildiğinde BB/3 simülatörü kapatılır.

3.1.2. Düzen menüsü

Düzen menüsünde Kes, Kopyala, Yapıştır, Sil ve Tümünü Sil seçenekleri vardır. Bu seçenekler yardımıyla ana hafızanın gösterildiği tablo içerisinde seçilen işlem yapılır. Hafıza hücrelerini seçmek için seçilecek hücre üzerinde farenin sol tuşuna tıklayıp fare sürüklenmelidir. Düzen menüsünde bulunan seçenekler, BB3 simülatörü ana hafızasının gösterildiği tablo üzerinde farenin sağ tuşuna tıklandığında da ortaya çıkar. Yapıştır seçeneği normalde aktif değildir, seçeneklerde görülmez. Yapıştır seçeneğinin aktif olması için daha önce kesme veya kopyalama işleminin yapılması gereklidir.

Kes seçeneği, seçimi yapılan BB/3 simülatörü ana hafıza kelime veya kelimelerinin muhtevalarını kesip bilgisayarın hafızasına alır.

Kopyala seçeneği, seçimi yapılan BB/3 simülatörü ana hafıza kelime veya kelimelerinin muhtevalarını kopyalarını bilgisayarın hafızasına alır.

Yapıştır seçeneği, daha önce Kes veya Kopyala ile bilgisayar hafızasına aktarılan BB/3 simülatörü ana hafıza kelimelerinin muhtevalarını, o an aktif olan ana hafıza kelimesinden itibaren yapıştırır.

Sil seçeneği, seçimi yapılan BB/3 simülatörü ana hafıza kelime veya kelimelerinin muhtevalarını siler, bu hafıza kelimelerine “0000” değeri aktarır.

Tümünü Sil seçeneği, BB/3 simülatörü ana hafıza kelimelerinin tümünün muhtevalarını siler ve tüm kelimelere “0000” değeri aktarır. Ayrıca tüm bit, bayrak ve kayıtlık değerlerini de silerek ilk değerlerine kurar.

3.1.3. Çalıştır menüsü

Çalıştır menüsü BB/3 simülatörünün ana hafızasında yazılı olan izlence örneklerinin yürütülüş tarzını belirler. BB/3 simülatöründe izlence yürütülürken beş izleniş tarzı vardır: Her zaman adımı bitiminde (Zaman-Zaman), her devir bitiminde (Devir-Devir), her komutun ifa devri bitiminde (Komut-Komut), serbest (izleniş yok) ve duraklı serbest(Durak noktalarında izleniş var). Bu izlence yürütülüş tarzları BB/3 simülatörü ana penceresindeki komut düğmeleri yardımıyla da seçilebilir.

Zaman-Zaman çalıştır seçeneği seçildiğinde, izlence bir zaman adımı çalışır, yani zaman_zaman_calistir altyordamı bir defa çağrılır.

Devir-Devir çalıştır seçeneği seçildiğinde, izlence bir devir çalışır, yani z[3]=1 olana kadar, zaman_zaman_calistir altyordamı çağrılır.

Komut-Komut çalıştır seçeneği seçildiğinde, izlence bir komut ifa edilene kadar çalışır, yani d[2]=1 ve z[3]=1 olana kadar, zaman_zaman_calistir altyordamı çağrılır.

Serbest çalıştır seçeneği seçildiğinde, izlence otomatik olarak çalışır. İzlence bitimine kadar zaman_zaman_calistir altyordamı çağrılır.

Duraklı Serbest çalıştır seçeneği seçildiğinde, izlence otomatik olarak çalışır. İzlence bitimine veya Ki'nin değeri bir durak noktasına eşit olana kadar zaman_zaman_calistir altyordamı çağrılır. Özel olarak incelenmek istenen hafıza kelimelerine durak noktası eklenir. Durak noktası eklemek istenilen hafıza kelimesinin üzerinde farenin sağ tuşuna tıklanır. Çıkan menüden "Durak Noktası Ekle" seçilir. Durak noktası eklenen hafıza kelimesinin rengi kırmızı olur. Durak noktası ekli bir hafıza kelimesinin durak noktasını kaldırmak için yine fare hafıza kelimesinin üzerinde iken farenin sağ tuşuna tıklanır. Çıkan menüden "Durak Noktasını Kaldır" seçilir.

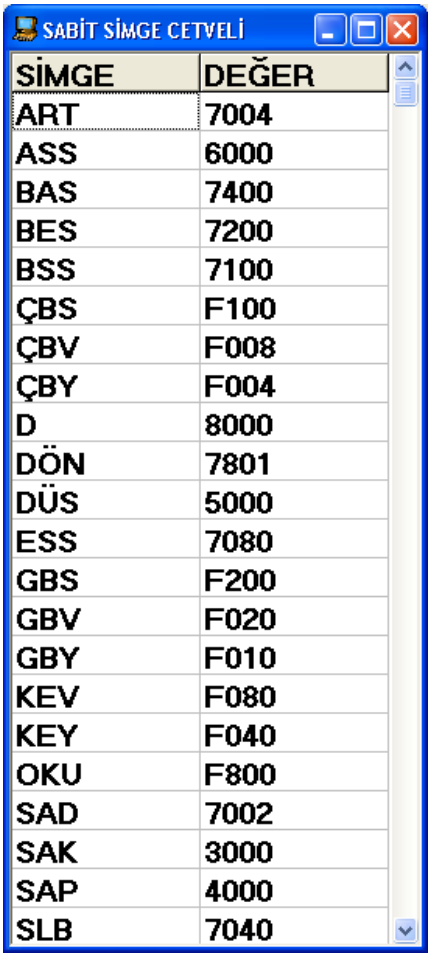
3.1.4. Araçlar menüsü

Araçlar menüsü BB/3 simülatörünü kullanırken yararlanılabilecek araçlardan oluşmaktadır. Araçlar menüsünde, Sayı Çevirici ve Sabit Simge Cetveli bulunmaktadır.

Sayı Çevirici seçeneği seçildiğinde, Sayı çevirici yardımcı program çalışır. Resim 3.2'de görülen bu program; değeri girilen ve tabanı seçilen bir sayının istenilen tabandaki karşılığını bulmayı sağlar. Taban olarak "2, 4, 8, 10, 16" değerleri seçilebilir.

Resim 3.2. Sayı çevirici yardımcı programı.

Sabit Simge Cetveli seçeneği seçildiğinde, BB/3 bilgisayar modelinin komutlarının listelendiği, Sabit simge cetveli penceresi açılır. Resim 3.3'de görülen bu pencerede BB/3 bilgisayar modeli komutlarının simgeleri ve değerleri alfabetik sırayla, bir tabloda gösterilir.



Resim 3.3. Sabit simge cetveli penceresi.

3.1.5. Yardım menüsü

Yardım menüsü; BB/3 simülatörü ve menüleri hakkında bilgi sahibi olunmayı sağlayan kullanıcı kılavuzuna ulaşmayı sağlar.

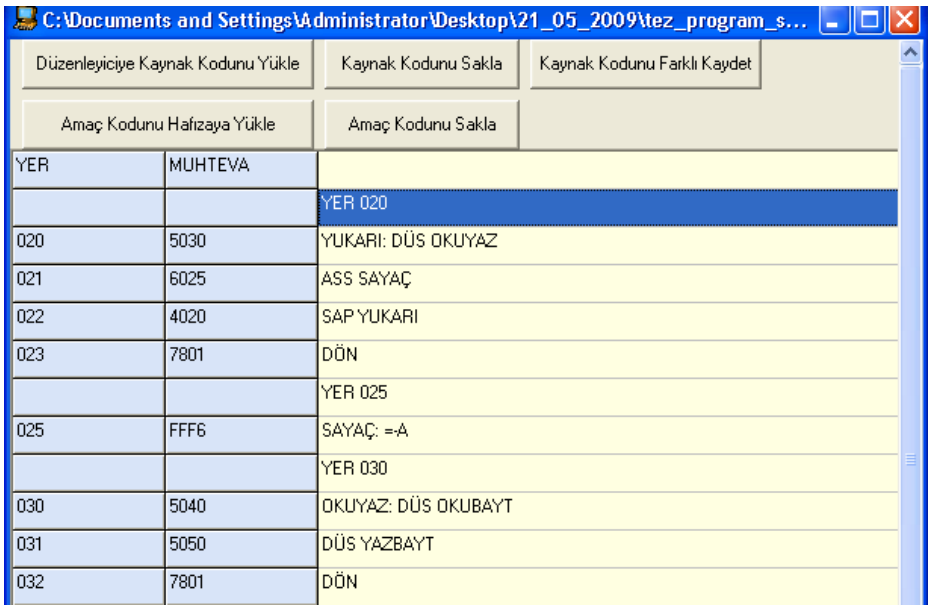
3.1.6. Birleştirici Komut Seçeneği

Birleştirici komut seçeneği BB/3 birleştirici penceresine ulaşmayı sağlar. BBSD simgesel dili ile yazılmış izlence kodlarına kaynak kod denir.

3.2. Birleştirici

BBSD dili kullanılarak BB/3 bilgisayar modelinde simgesel kod kullanılarak gerçek ve kapsamlı izlenceler yazılabilmesi için, BB/3 donanım simülatöründe bir birleştirici de mevcuttur. Bu birleştirici ciddi izlence örnekleri yazmada ve onları okumada kolaylık sağlar. Bu birleştirici Sabit simgeler (BB/3 komutları), tanımlanmış simgeler (Yaftalar), sabit değerler, değer ifadeleri, makrolar ve yardımcı komutlar kullanılarak, kaynak kodları girilen izlencelerin amaç kodlarını oluşturarak, BB/3 donanım simülatörünün ana hafızasına aktarmayı sağlar.

Birleştirici programı çalıştırıldığında ekrana Resim 3.4'deki pencere gelir. Bu pencerede izlencelerin yazılabilmesi için gerekli bir tablo bulunmaktadır. Tablo üç sütundan oluşmaktadır. BBSD dili şeklindeki izlence satırları en sağda bulunan sütuna yazılır. BBSD dili ile yazılmış olan izlencede hata yoksa birleştirici her satıra tekabül eden amaç kodunu "MUHTEVA" sütununa aktarır. BBSD dili ile yazılan izlence satırlarının ana hafızada yerleşecekleri yerleri "YER" sütununda belirtilir. Yalnızca yardımcı komutlardan oluşan satırın "MUHTEVASI" ve "YER" değeri belirtilmez.



Resim 3.4. Birleştirici penceresi.

3.2.1. Birleştiricinin çalışma prensipleri

Birleştirici BBSD'nin tanımına uygun olarak çalışmaktadır. BBSD dilinde birleştiriciye yazılan izlence, "YER" yardımcı komutuyla başlamalıdır. Bu sayede izlencenin, 4K olan ana hafızada yerleştirilmeye başlanacağı yer belirlenmiş olur. İzlencede birleştirilecek olan her satırın yer değeri vardır. Herhangi bir satırın YER değerini belirlemek için BBSD dilinin "YER" yardımcı komutu kullanılır. YER komutu kullanılmadığında, satırın yer değeri, bir önceki satırın yer değerinin bir fazlasıdır. YER değeri olarak "000" ile "FFF" arasında onaltılık tabanda bir sayı girilmelidir. Aksi halde hatalı bir giriş yapılmış olur. Sadece yardımcı komutlardan veya makrodan oluşan bir satırın "YER" değeri belirlenmez.

Birleştirici programında "/" yardımcı komutu yorum eklemek için kullanılır. "/" işaretinden sonra gelen ifadeler birleştirici tarafından değerlendirilmeye alınmaz. Birleştirici izlenceyi "SON" yardımcı komutu satırına kadar derler. "SON" yardımcı komutundan sonraki satırlar birleştirme işlemine tabi tutulmaz.

İzlencelerin makine diline çevrilişleri iki üç safhada olur. İlk safhada makrolar tanımlanır. İkinci safhada yaftaların adres değerleri belirlenerek "Değişken Simge Cetveli" oluşturulur. Değişken simge cetveli, izlencede kullanılan yaftaların isimleri ve işaret ettikleri adres değerlerini barındırır. Cetvel, yafta isimlerine göre alfabetik sırada oluşturulur. Simgesel kodla yazılan izlencelerde kullanılan yafta simgeleri bir harf ile başlayıp "," veya ":" işareti ile sonlanmalıdır. Bir yafta tek kelimeden oluşmalıdır ve herhangi bir satırda tanımlanan yafta simgesi başka bir satırda tekrar kullanılamaz. Aksi halde hata meydana gelir. Bir satırda sadece yafta tanımlanmış ise YER değeri artırılmaz. Üçüncü safhada ise birleştirme işi yapılır ve eğer hata yok ise, her satıra tekabül eden amaç kodu ve yer değeri oluşturulur.

Birleştirici penceresinde beş farklı işlem yapılabilmektedir. Bu işlemlere komut düğmeleri yardımıyla ulaşılabilir. Bunlar; "Düzenleyiciye Kaynak Kodunu Yükle",

“Kaynak Kodunu Sakla”, “Kaynak Kodunu Farklı Kaydet”, “Amaç Kodunu Hafızaya Yükle” ve “Amaç Kodunu Sakla” işlemleridir.

3.2.2. Düzenleyiciye kaynak kodunu yükle

Düzenleyiciye Kaynak Kodunu Yükle seçeneği, BBSD ile yazılıp daha önce bilgisayara kaydedilmiş olan izlencelerin seçilip, bu izlenceleri birleştiriciye aktarılmasını sağlar. Simgesel dil dosyalarının uzantıları “bsd” dir.

3.2.3. Kaynak kodunu sakla

Kaynak Kodunu Sakla seçeneği, Birleştirici penceresinde simgesel kodları yazılmış olan izlenceyi “bsd” uzantılı olarak bilgisayara kaydetmeyi sağlar.

3.2.4. Kaynak kodunu farklı kaydet

Kaynak Kodunu Farklı Kaydet seçeneği, Birleştirici penceresinde simgesel kodları yazılmış olan izlenceyi “bsd” uzantılı olarak farklı bir konuma farklı bir isimle kaydetmeyi sağlar.

3.2.5. Amaç kodunu hafızaya yükle

Amaç Kodunu Hafızaya Yükle seçeneği, Birleştirici penceresinde simgesel kodları yazılmış olan izlencenin amaç kodlarını BB/3 simülatörünün ana hafızasına yüklemeyi sağlar. Eğer izlence satırının herhangi birinde hata varsa amaç kodu simülatörün ana hafızasına aktarılamaz. İzlencede herhangi bir hata yoksa, her satırın muhtevası, yer sütunundaki adrese eşit ana hafıza kelimesine aktarılır.

3.2.6. Amaç kodunu hafızaya yükle

Amaç Kodunu Hafızaya Yükle seçeneği, Birleştirici penceresinde simgesel kodları yazılmış olan izlencenin amaç kodlarını “bb3” uzantılı bir dosya olarak bilgisayara kaydetmeyi sağlar.

4. DENEYLER

Hazırlanan simulatör ve birleştiricinin çalışmasını kontrol etmek için deney ve örnekler hazırlanmıştır. Bu deneylerin ve örnekler simulatör ve birleştiriciye uygulanarak, çalıştırılabilirlikleri izlenmiş ve elde edilen sonuçlara göre yazılımda düzeltmeler yapılmıştır.

Deney1: Küçük bir izlence yapılarak giriş biriminden 10 tane karakteri yazdırıp (Bir döngü içerisinde OKUYAZ altyordamı kullanılacak) duran bir deney programı yazınız [1].

Hedef : İzlence güdümlü giriş-çıkış işlemlerinin hatasız yapıldığını göstermek.

Cevap:

Yer	Muhteva	Yorum
020 YUKARI:	5030	DÜS OKUYAZ
021	6025	ASS SAYAÇ
022	4020	SAP YUKARI (020)
023	7801	DÖN
025 SAYAÇ	FFF6	SAYAÇ= -10
030 OKUYAZ:	5040	DÜS OKUBAYT
031	5050	DÜS YAZBAYT
032	7801	DÖN
040 OKUBAYT Y1	F200	GBS
041	4040	SAP Y1 (40)
042	F800	OKU
043	7801	DÖN
050 YAZBAYT Y2	F100	ÇBS
051	4050	SAP Y2 (50)
052	F400	YAZ
053	7801	DÖN

Deney2: Modern bilgisayarlarda giriş-çıkış işlemleri yapılırken bayrak kontrolünden kaynaklanan mahzurları ortadan kaldıran bir örnektir [1].

Hedef: İzlence güdümlü giriş-çıkış işleminde sürekli bayrak yoklamadan kaynaklanan zaman kaybını önlemek. Giriş birimi ile çıkış birimi birbirinden

bağımsız çalışan donanım cihazlarıdır. Bu nedenle farklı hızlarda çalışırlar. Giriş işlemi yapılırken olabilecek veri kayıplarını önlemek için hafızanın bir kısmını tampon olarak kullanmak.

Cevap:

Yer	Muhteva	Yorum
000 KESBAKYOR:	300E	SAK Bİ (00E)
001	7002	SAD
002	300F	SAK E (00F)
003 GİRİŞİ YOKLA:	F200	GBS
004	4006	SAP ÇIKIŞI YOKLA (006)
005	5010	DÜS GİRİŞE BAK (010)
006 ÇIKIŞI YOKLA:	F100	ÇBS
007	4009	SAP SIRADAKİ (009)
008	5015	DÜS ÇIKIŞA BAK (015)
009 SIRADAKİ:	200F	YÜK E (00F)
00A	7001	SOD
00B	200E	YÜK Bİ (00E)
00C	F080	KEV
00D	7801	DÖN (KESBAKYOR)
00E Bİ:	0000	00E=0 (Bİ)
00F E:	0000	00F=0 (E)
010 GİRİŞE BAK:	F800	OKU
011	F008	ÇBV
012	603C	ASS OKUNAN (03C)
013	B03C	D SAK OKUNAN (03C)
014	7801	DÖN (GİRİŞE BAK)
015 ÇIKIŞA BAK:	203C	YÜK OKUNAN (03C)
016	5023	DÜS TAM2 (023)
017	103D	TOP YAZILAN (03D)
018	7200	BES
019	401E	SAP BEKLE (01E)
01A	603D	ASS YAZILAN (03D)
01B	A03D	D YÜK YAZILAN (03D)
01C	F400	YAZ
01D DÖNÜŞ:	7801	DÖN (ÇIKIŞA BAK)
01E BEKLE:	F004	ÇBY
01F	401D	SAP DÖNÜŞ (01D)
020 HEMEN_YÜKLE:	7804	YDY

021	7802	Yİİ
022	7801	DÖN (HEMEN YÜKLE)
023 TAM2:	7010	TMB
024	7004	ART
025	7801	DÖN (TAM2)
030 ANA SÜREÇ:	5020	DÜS HEMEN YÜKLE (020)
031	0050	Giriş/Çıkış Destek Alanı Başlang. (050)
032	303C	SAK OKUNAN (03C)
033	303D	SAK YAZILAN (03D)
034	F080	KEV
035 TEKRAR:	5020	DÜS HEMEN YÜKLE (020)
036	FF41(-0BF)	Giriş/Çıkış Destek Alanı Bitiş (0BF)
037	103D	TOP YAZILAN (03D)
038	7400	BAS
039	4035	SAP TEKRAR (035)
03A	F040	KEY
03B	7801	DÖN (İzlencenin Mantıksal Sonu)
03C OKUNAN:	0000	03C=000 (OKUNAN İŞARETÇİSİ)
03D YAZILAN:	0000	03D=000 (YAZILAN İŞARETÇİSİ)

Aşağıda; BB/3 bilgisayar modelinin simülatör ve birleştiricisinde altyordamların ve yığıt işlemlerinin çalışmasını kontrol eden örnekler yer almaktadır.

Örnek1: Öyle bir altyordam yazınız ki DÜS "ELEMAN_SAYISI" diye çağrılsın, bu çağıran komuttan hemen sonra bir veri kelimesi gelsin, altyordam bu veri kelimesini dizinin izafi başlangıç adresi olarak alarak dizinin kaç elemanlı olduğunu bulup bu değeri birikece yükledikten sonra dursun.

Cevap:

```
YER 010

DÜS DÖN4SAĞA

=50

DÖN

YER 40
```

```
SAK ADRES
DÖN4SAĞA, SAD
DÖNGÜ, TOP BAS_ADRES
SAK E
D YÜK ADRES
SAD
BSS
SAD
SAP BİTİR
SAD
DÜS BAYTDEĞİŞ
SAK GEÇİCİ
BİTİR: YÜK DÜĞÜM_SAYISI
SAD
BAS_ADRES, =0A00
VE SF000
SFF00, =FF00
SAK RAKAM
YER 035
YÜK GEÇİCİ
DÜS ELEMAN_SAYISI
```

```
VE S0FFF
DÖN
TOP RAKAM
ELEMAN_SAYISI, YOA
SAK GEÇİCİ
D YÜK ADRES
YÜK E
SAK ADRES
SOD
VE SFF00
YÜK GEÇİCİ
SAP DEVAM
DÖN
DEVAM: ASS DÜĞÜM_SAYISI
E, =0
SAP DÖNGÜ
GEÇİCİ, =0
DÖN
RAKAM, =0
ADRES, =0
SF000, =F000
```

```
DÜĞÜM_SAYISI, =0

S0FFF, =0FFF

BAYTDEĞİŞ, DÜS DÖN4SAĞA
```

Örnek2: Öyle bir altyordam yazınız ki DÜS "DEĞER_TOPLA" diye çağrılsın. Bu çağıran komuttan hemen sonra bir veri kelimesi gelsin, altyordam bu veri kelimesini dizinin izafi başlangıç adresi olarak alarak dizinin tüm düğümlerinin ASCII kod değerlerini toplayıp bu toplam değerini birikece yükledikten sonra dursun.

Cevap:

```
YER 010
DÜS DEĞER_TOPLA
=50
DÖN
YER 20
DEĞER_TOPLA, YOA
SAK ADRES
D YÜK ADRES
DÖNGÜ, TOP BAS_ADRES
SAK ADRES
D YÜK ADRES
VE SFF00
BSS
SAP DEVAM
SAP BİTİR
DEVAM: SAK GEÇİCİ
D YÜK ADRES
VE S00FF
TOP TOPLAM
SAK TOPLAM
```

```
YÜK GEÇİCİ
DÜS BAYTDEĞİŞ
SAP DÖNGÜ
BİTİR: YÜK TOPLAM
DÖN
BAS_ADRES, =0A00
ADRES, =0
SFF00, =FF00
S00FF, =00FF
TOPLAM, =0
YER 03A
BAYTDEĞİŞ, DÜS DÖN4SAĞA
DÜS DÖN4SAĞA
DÖN
YER 40
DÖN4SAĞA, SAD
SAK E
SAD
SAD
SAD
SAK GEÇİCİ
SAD
VE SF000
SAK RAKAM
YÜK GEÇİCİ
VE S0FFF
TOP RAKAM
SAK GEÇİCİ
YÜK E
SOD
YÜK GEÇİCİ
DÖN
```

```
E, =0
GEÇİCİ, =0
RAKAM, =0
SF000, =F000
S0FFF, =0FFF
```

Örnek3: Öyle bir altyordam yazınız ki DÜS "ŞEKİLCİK_SAY" diye çağrılsın. Bu çağıran komuttan hemen sonra iki veri kelimesi gelsin, altyordam birinci veri kelimesini dizinin izafi başlangıç adresi, ikinci veri kelimesini ise bir şekilciğin ASCII kodu olarak alarak, başlangıç izafi adresinden itibaren bu şekilcikten bağlı diziden kaç tane olduğunu bularak bulup bu değeri birikece yükledikten sonra dursun [1].

Cevap:

```
YER 010
DÜS ELEMAN_SAYISI
=50
=41
DÖN
YER 20
ELEMAN_SAYISI, YOA
SAK ADRES
YOA
SAK GEÇİCİ
D YÜK GEÇİCİ
SAK ASCII_KOD
D YÜK ADRES
DÖNGÜ, TOP BAS_ADRES
SAK ADRES
D YÜK ADRES
VE SFF00
```

```
BSS
SAP DEVAM
SAP BİTİR
DEVAM: DÜS BAYTDEĞİŞ
SAK GEÇİCİ
D YÜK ADRES
VE S00FF
TMB
ART
TOP ASCII_KOD
BSS
SAP GİT
ASS ŞEKİLCİK_SAYISI
GİT, YÜK GEÇİCİ
SAP DÖNGÜ
BİTİR: YÜK ŞEKİLCİK_SAYISI
DÖN
BAS_ADRES, =0A00
ADRES, =0
ASCII_KOD, =0
SFF00, =FF00
S00FF, =00FF
ŞEKİLCİK_SAYISI, =0
YER 015
BAYTDEĞİŞ, DÜS DÖN4SAĞA
DÜS DÖN4SAĞA
DÖN
YER 50
DÖN4SAĞA, SAD
SAK E
SAD
SAD
```

```
SAD
SAK GEÇİCİ
SAD
VE SF000
SAK RAKAM
YÜK GEÇİCİ
VE S0FFF
TOP RAKAM
SAK GEÇİCİ
YÜK E
SOD
YÜK GEÇİCİ
DÖN
E, =0
GEÇİCİ, =0
RAKAM, =0
SF000, =F000
S0FFF, =0FFF
```

Örnek4: Fibonacci dizisinin n. terimini hesaplayan bir altyordam [1]:

```
F(n)= { n, ……F(n-1)+ F(n-2)
eğer n=0 veya n=1 ise
aksi halde
Yordam: Fib(n)
Eğer n=0 veya n=1 ise «Fib← n; dön»
Fib← Fib(n-1) + Fib(n-2)
Dön
Son

YER 100
YÜK SAYI
```

```
DÜS Fib
DÖN
SAYI, =7
YER 110
/ Fibonacci(n) değerini hesaplamak için tekrar kendini çağırış
/ tekniğinin kullanıldığı bir altyordam
Fib, / n değerini Bİ -den alır, Fib(n) değerini Bİ -de bırakarak döner.
DÜS BiSıfDeğSek/ Bİ= n olduğundan, n≠ 0 ise, sıradaki komutu atla.
DÖN / n= 0 olduğuna göre, Fib(n)= 0 olarak dön.
DÜS EksiltBİ / Bİ← n-1
BSS / n-1 bulunan Bİ= 0, yani n= 1 ise, sıradaki komutu atla.
SAP Y1 / Bİ= n-1≠ 0 ise, Y1 -e git.
ART / Bİ← 1
DÖN / n= 1 olduğuna göre, Fib(n)= 1 olarak dön.
Y1, YVK / YığıtaKoy: (n-1)
DÜS Fib / (n-1) değeriyle tekrar Fib -i çağır
SAK Fib1/ Fib -den dönüşte Bİ -de olan Fib(n-1) değerini sakla.
YVA / Y1 -de yığıta saklanan (n-1) değerini Bİ -ye aktar.
SAK BİH / BİH← (n-1)
YÜK Fib1/ Bİ← Fib(n-1)
Y2, YVK / YığıtaKoy: Fib(n-1)
YÜK BİH / Bİ← (n-1)
DÜS EksiltBİ / Bİ← (n-2)
DÜS Fib / (n-2) değeriyle tekrar Fib -i çağır
SAK Fib2/ Fib -den dönüşte Bİ -de olan Fib(n-2) değerini sakla.
YVA / Y2 -de yığıta saklanan Fib(n-1) değerini Bİ -ye aktar.
TOP Fib2/ Bİ= Fib(n-1)+ Fib(n-2) olur.
DÖN
BİH, =0 / Bİ -yi saklamak için hafıza yeri.
Fib1, =0 / Fib(n-1) değerinin saklandığı yer.
Fib2, =0 / Fib(n-2) değerinin saklandığı yer.
BiSıfDeğSek, / Bİ≠ 0 ise, sıradaki komutu seken altyordam.
```

```
BSS
Yİİ
DÖN
EksiltBİ, / Bİ -yi bir eksilten altyordam
TMB ART
TMB
DÖN
SON
```

5. SONUÇ VE ÖNERİLER

Bu çalışmada Delphi 7.0 programlama dili kullanılarak BB/3 bilgisayarı için bir donanım simülatörü ve BBSD dilini kullanabilmek için bir birleştirici oluşturulmuştur.

BB/3 simülatörü bir öğretim dönemi boyunca 100 kişilik bir öğrenci grubunda denenmiştir. Öğrenci grubu BB/3 simülatör ve birleştiricisini kullanarak çeşitli deneyler gerçekleştirmişlerdir. Bu simülatörün etkisi altında öğrencilerin son derece motive edildikleri ve konuları daha kolay kavradıkları gözlemlenmiştir.

Simülatör açık kaynak kodludur, böylece bu simülatörü geliştirmek isteyecekler açısından yazılımın sistematik olmasına dikkat edilmiştir. Simülatör sayesinde, öğrenciler simülatör kodlarını inceleyip anlayabilirler ve hatta donanım üzerinde yapmak istedikleri değişiklikleri simülatöre uyarlayıp, simülatörü değiştirebilirler. Bununla birlikte BB/3 simülatörün kodlarında değişiklikler yaparak, kullanıcılar yeni oluşturdukları komutları ekleyebilir veya var olan komutların işleyişlerini değiştirebilirler. Simülatörde ayrıca, öğrencilerin sayı tabanlarını daha kolay kavramaları için bir sayı çevirici yer almaktadır.

Birleştirici sayesinde; BBSD dili ile BB/3 bilgisayar modelinde simgesel kod kullanılarak gerçek ve kapsamlı izlenceler yazılabilir. Bu birleştirici sayesinde öğrenciler izlencenin kaynak kodlarını (Simgesel kod) ve amaç kodlarını (Makine kodu) irdeleyebilmektedir ve böylece dil teorisi hakkında bilgi sahibi olmaktadır. Ayrıca birleştirici kullanarak BBSD dili ile izlenceler yazarken, öğrencilere *assembler* ilkelerinden söz etmek de mümkün olmaktadır.

BB/3 simülatörü sayesinde Bizim Bilgisayarın daha önceki modellerinde olmayıp da BB/3 olarak eğitim müfredatına alınan modeldeki yığıt yeniliğinin doğruluğu örneklerle kanıtlanabilmiştir.

Bu yazılım,

- Simgesel dilin nasıl makine diline çevrildiği
- Birleştirme dilinin temellerini öğretir
- Yığıt yapısının nasıl çalıştığı
- Merkezi işlem mimarisini (kayıtlık, bayraklar, bitler, giriş çıkış işlemleri, giriş çıkış aygıtlarının koordinasyonu, kesinti yapıları) öğrenilir.
- Kayıtlık aktarım dili

Öğretilmesine yardımcı olabilir.

Donanım tasarımı olan Bizim Bilgisayarın farklı modelleri oluşturularak aynı simülatör ve birleştirici yeni oluşturulacak donanım tasarımlarına uyarlanabilir.

KAYNAKLAR

1. İnternet: Gazi Üniversitesi "Donanım Tasarımı Dersi, Ders Notları" http://w3.gazi.edu.tr/~dcalik/=Duyuru=/Duyuru=DERSLER/duyuru-Lisans/d-lisans.htm (2009).

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. Stabler, E. P., "Microprogram Transformations", ***IEEE Transactions On Computers***, 19(10): 909 (1970).

4. Schorr, H., "Computer-Aided Digital System Design and Analysis Using a Register Transfer Language", ***IEEE Transactions,*** 13: 731 (1964).

5. Khan, F., Anwar, S., "Design and Construction of a PC-Based Stack Machine Simulator for Undergraduate Computer Science & Engineering Courses", ***Frontiers in Education Conference***, USA, 1234 (1997).

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

7. Donovan, J. J., "System Programming" ***McGraw-Hill International Book Co.,*** Singapur, 6-7, 62 (1985).

8. Chisolm, K. C., Lisonbee, J. C., "The Use if Computer Language Compiler In Legacy Code Migration" ***IEEE Systems Readiness Technology Conferenc*e,** USA, 140 (1999).

EKLER

EK-1. BB/3 Simülatör Kodları

```
procedure TForm1.zaman_zaman_calistir();
{bb3 izlencesini zaman-zaman çalıştırır.}
var
 KEST, KI_Say, E_ :Bit; {Elde biti için tampon değişken}
 i,p,q: byte;
 t,r: array [0..3] of bit;
 begin
 if (C=0) and (Print_suresi>0) then
    print_sureci
 else if (C=0) and (Print_suresi=0) then
  begin
     timer1.Enabled :=False;
     exit;
  end
 else if C=1 then
 begin   {Ekrandaki Text Kutucuklarından Kayıtlıklara Değer Aktarılıyor}
  CALIS_Text.Text:=inttostr(C);
  KI_Say:=0;
  KI_G:=strtoint(onluga_cevir(KI_Text.text,16));
  KI_C:=strtoint(onluga_cevir(KI_Text.text,16));
  KK:=strtoint(onluga_cevir(KK_Text.text,16));
  HAK_C:=strtoint(onluga_cevir(HAK_Text.text,16));
  HDK_C:=strtoint(onluga_cevir(HDK_Text.text,16));
  E:=strtoint(ELDE_Text.text);
  BI_G:=strtoint(onluga_cevir(BIRIKEC_Text.text,16));
  BI_C:=strtoint(onluga_cevir(BIRIKEC_Text.text,16));
  GIRB:=strtoint(GIRB_Text.text);
  GIRK:=strtoint(onluga_cevir(GIRK_Text.text,16));
  CIKB:=strtoint(CIKB_Text.text);
  CIKK:=strtoint(onluga_cevir(CIKK_Text.text,16));
  KYV_C:=strtoint(KYV_Text.text);
  GBI:=strtoint(GBI_Text.text);
  CBI:=strtoint(CBI_Text.text);
  YI:=strtoint(onluga_cevir(YI_Text.text,16));
  A:=strtoint(A_Text.caption);
  B:=strtoint(B_Text.caption);
```

EK-1. (Devam) BB/3 Simülatör Kodları

```
For i:=0 To 3 do Begin D[i]:=0;Z[i]:=0; End;
D[2*B+A]:=1;            {AB Kod Çözücü }
Z[SS]:=1;               {SS Kod Çözücü }

For i:=0 To 7 do K[i]:=0;
K[(KK and $7000) SHR 12]:=1;              {İŞLEM KODU Kod Çözücü}
For i:=15 downto 0 do  V[i]:=(KK SHR i) and $0001; {KK nın  bitleri}

KEST:=((GBI And GIRB) Or (CBI And CIKB));//Kesinti talep girişi bağlantısı.
//***** GETİR DEVRİ *****//
  if D[0]*Z[0]=1 Then HAK_G:=KI_C;
  if D[0]*Z[1]=1 Then
     Begin
       HDK_G:=strtoint(onluga_cevir(HAFIZA[HAK_C],16));
       KI_Say:=1;
     End;
  if D[0]*Z[2]=1 Then
     Begin
       KK:=HDK_C;
       HAK_G:=YI;
     End;
  if (1-K[7])*V[15]*D[0]*Z[3]=1 Then A:=1;
  if (1-((1-K[7])*V[15]))*D[0]*Z[3]=1 Then B:=1;

  //***** DOLAYLI DEVRİ *****//
  if D[1]*Z[0]=1 Then HAK_G:=HDK_C and $0FFF;
  if D[1]*Z[1]=1 Then HDK_G:=strtoint(onluga_cevir(HAFIZA[HAK_C],16));
  if D[1]*Z[3]=1 Then Begin B:=1;A:=0;End;

  //***** KESİNTİ DEVRİ *****//
  if D[3]*Z[0]=1 Then
        Begin YI:=YI-1;HDK_G:=(HDK_G and $F000) or KI_C; KI_G:=0; End;
  if D[3]*Z[1]=1 Then HAK_G:=YI;
  if D[3]*Z[2]=1 Then Begin HAFIZA[HAK_C]:=inttohex(HDK_C,4);KYV_G:=0;end;
  if D[3]*Z[3]=1 Then Begin A:=0;B:=0;End;
```

EK-1. (Devam) BB/3 Simülatör Kodları

```
//***** İFA DEVRİ ORTAK SON *****//
      if D[2]*Z[3]=1 Then
       begin
        if (KYV_C And KEST) = 1 Then A:= 1   //KEST
        Else B:= 0;
       end;

  {-----Hafıza Adresleyen Komutlar-----------------------------------}
{VE }      if K[0]*D[2]*Z[0]=1 Then HAK_G:=HDK_C and $0FFF;
         if K[0]*D[2]*Z[1]=1 Then HDK_G:=strtoint(onluga_cevir(HAFIZA[HAK_C],16));
         if K[0]*D[2]*Z[2]=1 Then BI_G:=BI_C and HDK_C;
{TOP}    if K[1]*D[2]*Z[0]=1 Then HAK_G:=HDK_C and $0FFF;
         if K[1]*D[2]*Z[1]=1 Then HDK_G:=strtoint(onluga_cevir(HAFIZA[HAK_C],16));
         if K[1]*D[2]*Z[2]=1 Then
            Begin BI_G:=BI_C+HDK_C; if BI_G<HDK_C Then E:=1 Else E:=0; end;
{YUK}      if K[2]*D[2]*Z[0]=1 Then begin HAK_G:=HDK_C and $0FFF; BI_G:=0; end;
         if K[2]*D[2]*Z[1]=1 Then
         begin HDK_G:=strtoint(onluga_cevir(HAFIZA[HAK_C],16)); BI_G:= not BI_C; end;
         if K[2]*D[2]*Z[2]=1 Then BI_G:=(BI_C and HDK_C);
{SAK}      if K[3]*D[2]*Z[0]=1 Then HAK_G:=HDK_C and $0FFF;
         if K[3]*D[2]*Z[1]=1 Then HDK_G:=BI_C;
         if K[3]*D[2]*Z[2]=1 Then HAFIZA[HAK_C]:=inttohex(HDK_C,4);
{SAP}      if K[4]*D[2]*Z[0]=1 Then KI_G:=HDK_C and $0FFF;
{DUS}      if K[5]*D[2]*Z[0]=1 Then begin HDK_G:=(HDK_G and $F000) or KI_C;
KI_G:=(HDK_C and $0FFF);YI:=YI-1; end;
         if K[5]*D[2]*Z[1]=1 Then HAK_G:=YI;
         if K[5]*D[2]*Z[2]=1 Then HAFIZA[HAK_C]:=inttohex(HDK_C,4);
{ASS}      if K[6]*D[2]*Z[0]=1 Then HAK_G:=HDK_C and $0FFF;
         if K[6]*D[2]*Z[1]=1 Then HDK_G:=strtoint(onluga_cevir(HAFIZA[HAK_C],16));
         if K[6]*D[2]*Z[2]=1 Then HDK_G:=HDK_C+1;
         if K[6]*D[2]*Z[3]=1 Then
         Begin HAFIZA[HAK_C]:=inttohex(HDK_C,4); if HDK_C=0 Then KI_SAY:=1;End;

  {-----Kayıtlık Komutları-------------------------------------}
   p:=(1-V[15])*K[7]*(1-V[11])*D[2];
   for i:=0 to 3 do
```

EK-1. (Devam) BB/3 Simülatör Kodları

```
r[i]:= p*z[i];
    begin
    {BAS}     if V[10]*r[0]=1 Then if (BI_G And $8000)=0 Then KI_SAY:=1;
    {BES}     if V[9]*r[0]=1 Then if (BI_G And $8000)=$8000 Then KI_SAY:=1;
    {BSS}     if V[8]*r[0]=1 Then if BI_G=0 Then KI_SAY:=1;
    {ESS}     if V[7]*r[0]=1 Then if E =0 Then KI_SAY:=1;
    {SLB}     if V[6]*r[0]=1 Then BI_G:=0;
    {SLE}     if V[5]*r[0]=1 Then E:=0;
    {TMB}      if V[4]*r[1]=1 Then BI_G:=not BI_C;
    {TME}      if V[3]*r[1]=1 Then E:=1-E;
    {ART}      if V[2]*r[2]=1 Then
                  Begin BI_G:=BI_C+1; if BI_G=0 Then E:=1 Else E:=0; End;
    {SAD}      if V[1]*r[3]=1 Then
                  Begin E_:=BI_C and $0001; BI_G:=(BI_C SHR 1)+E*$8000; E:=E_; End;
    {SOD}      if V[0]*r[3]=1 Then
                  Begin E_:=BI_C div $8000; BI_G:=(BI_C SHL 1)+E; E:=E_; End;
    end;

{-----Giriş-Çıkış Komutları----------------------------------}
      S:= V[15]*k[7]*D[2]*Z[3];
{OKU}     if V[11]*S=1 Then Begin BI_G:=(BI_C and $FF00) OR GIRK ; GIRB:=0; End;
{YAZ}     if V[10]*S=1 Then
            Begin
                CIKK:=BI_C and $00FF;
                CIKB:=0;
                Print_suresi:=100;
            end;
{GBS}     if V[9]*S =1 Then  if GIRB=1 Then KI_Say:=1;
{CBS}     if V[8]*S =1 Then  if CIKB=1 Then KI_Say:=1;
{KEV}     if V[7]*S =1 Then  KYV_G:=1;
{KEY}     if V[6]*S =1 Then  KYV_G:=0;
{GBV}     if V[5]*S =1 Then  GBI:=1;
{GBY}     if V[4]*S =1 Then  GBI:=0;
{ÇBV}     if V[3]*S =1 Then  CBI:=1;
{ÇBY}     if V[2]*S =1 Then  CBI:=0;
```

EK-1. (Devam) BB/3 Simülatör Kodları

```
{-----Yığıt Komutları----------------------------------}
 q:= (1-V[15])*K[7]*V[11]*D[2];
 for i:=0 to 3 do
    t[i]:= q*z[i];
 {YİE} if v[6]*t[0]=1 then YI:=YI-1;
 {YİA} if v[5]*t[2]=1 then YI:=YI+1;
 {YVK} if v[4]*t[0]=1 then YI:=YI-1;
      if v[4]*t[1]=1 then begin HAK_G:=YI;HDK_G:=BI_C; end;
      if v[4]*t[2]=1 then HAFIZA[HAK_C]:=inttohex(HDK_G,4);
 {YVA} if v[3]*t[0]=1 then BI_G:=0;
      if v[3]*t[1]=1 then
begin HDK_G:=strtoint(onluga_cevir(HAFIZA[HAK_C],16));BI_G:=not BI_C; end;
      if v[3]*t[2]=1 then begin BI_G:=(BI_C and HDK_C);YI:=YI+1; end;
{YDY} if v[2]*t[0]=1 then HDK_G:=strtoint(onluga_cevir(HAFIZA[HAK_C],16));
      if v[2]*t[1]=1 then begin HAK_G:=HDK_C and $0FFF;BI_G:=0; end;
      if v[2]*t[2]=1 then
begin  HDK_G:=strtoint(onluga_cevir(HAFIZA[HAK_C],16));BI_G:=not BI_C; end;
      if v[2]*t[3]=1 then BI_G:=(BI_C and HDK_C);
 {Yİİ} if v[1]*t[0]=1 then HDK_G:=strtoint(onluga_cevir(HAFIZA[HAK_C],16));
      if v[1]*t[1]=1 then HDK_G:=HDK_C+1;
      if v[1]*t[2]=1 then HAFIZA[HAK_C]:=inttohex(HDK_C,4);
 {DÖN} if v[0]*t[0]=1 then HDK_G:=strtoint(onluga_cevir(HAFIZA[HAK_C],16));
      if v[0]*t[1]=1 then KI_G:=(HDK_C and $0FFF);
      if v[0]*t[2]=1 then YI:=YI+1;
      if v[0]*t[3]=1 then
          if ((YI and $800) SHR 11 =0) then
               C:=0;  //YI'nın bit 11'i 0 ise Dur, Ç=0 olacak

   YI:=YI mod 4096 ;{YI FFF iken 1 artırılırsa 000 oluyor}
   if YI_Hucre<>YI then YI_Renklendir; //YI değiştiyse renklendiriliyor

   KI_C:=(KI_G+KI_SAY) mod 4096 ;{KI_SAY biti 1 ise KI bir ilerletiliyor}
   if KI_Hucre<>KI_C then KI_Renklendir; //KI değiştiyse renklendiriliyor

   {Kayıtlıkların Girişine Gelen Değerler Çıkışa Aktarılıyor}
   HDK_C:=HDK_G;
```

EK-1. (Devam) BB/3 Simülatör Kodları

```
            HAK_C:=HAK_G;
            BI_C:=BI_G;
            KYV_C:=KYV_G;
            Print_Sureci; {Yazma işlemi kontrolü için Altyordama gidiliyor}

            {Kayıtlık Değerleri Ekrandaki Text Kutucuklarına Aktarılıyor}
            KI_Text.Text:=  inttohex(KI_C,3);
            KK_Text.Text:= inttohex(KK,4);
            HAK_Text.Text:= inttohex(HAK_C,3);
            HDK_Text.Text:= inttohex(HDK_C,4);
            DOLAYLI_Text.Text:= inttostr((KK and $8000) SHR 15);
            ISK_Text.Text:= inttostr((KK and $7000) SHR 12);
            BIRIKEC_Text.Text:= inttohex(BI_C,4);
            ELDE_Text.Text:= inttostr(E);
            GIRB_Text.Text:=inttostr(GIRB);
            CIKB_Text.Text:= inttostr(CIKB);
            GIRK_Text.Text:= inttohex(GIRK,2);
            CIKK_Text.Text:= inttohex(CIKK,2);
            KYV_Text.Text:= inttostr(KYV_C);
            GBI_Text.Text:= inttostr(GBI);
            CBI_Text.Text:= inttostr(CBI);
            YI_Text.Text:=inttohex(YI,3);

            if (C=0) and (Print_suresi=0) then
               Timer1.Enabled:=false;
            CALIS_Text.Text:=inttostr(C);
            A_Text.Caption:= inttostr(A);
            B_Text.Caption:= inttostr(B);
            DEVIR_Text.Text:= inttostr(2 * B + A);
            SS:= (SS+1) mod 4;
            ZAMAN_Text.Text:= inttostr(SS);
            {Hafıza değerleri değiştiğinde ekranda hafızayı gösteren gridin değerleri güncelleniyor}
            if ((1-z[0])*(1-z[1])*D[2]*(1-K[7])) or (z[2]*D[2]*K[7]*V[11])=1 then
              Hafizaya_al();
          end;
end;
```

EK-1. (Devam) BB/3 Simülatör Kodları

```
procedure TForm1.Print_sureci();
{Print sürecini başlatır. Donanım gecikmesini devreye sokar}
 begin
  if Print_suresi >0 then
     begin
      Print_suresi:=Print_suresi-1;
      if (C=0) and (Print_suresi=0) then // C=0 olduktan sonra yazıcıdaki işi bitirmesi için
       begin
        CIKB:=1;
        if satir_basi then
          begin
            ListBox1.Items.Strings [ekran_satir]:=chr(CIKK);
            satir_basi:=false;
          end
        else
            ListBox1.Items.Strings [ekran_satir]:=ListBox1.Items.Strings
[ekran_satir]+chr(CIKK);
         end;
     end
  else if (CIKB=0) then
    begin
      CIKB:=1;
      if satir_basi then
        begin
          ListBox1.Items.Strings [ekran_satir]:=chr(CIKK);
          satir_basi:=false;
        end
      else
        ListBox1.Items.Strings [ekran_satir]:=ListBox1.Items.Strings
[ekran_satir]+chr(CIKK);
    end;
end;
```

Sabit Simge Cetveli Kodları

EK-1. (Devam) BB/3 Simülatör Kodları

```
procedure TForm2.SSimge_Cetveli_Olustur();
begin
Rewrite(Ssimge_cetveli);
SSC.kod:='7004';SSC.simge:='ART';seek(Ssimge_cetveli,filesize(Ssimge_cetveli));Write(Ssimge_cetveli,SSC);
SSC.kod:='6000';SSC.simge:='ASS';seek(Ssimge_cetveli,filesize(Ssimge_cetveli));Write(Ssimge_cetveli,SSC);
SSC.kod:='7400';SSC.simge:='BAS';seek(Ssimge_cetveli,filesize(Ssimge_cetveli));Write(Ssimge_cetveli,SSC);
SSC.kod:='7200';SSC.simge:='BES';seek(Ssimge_cetveli,filesize(Ssimge_cetveli));Write(Ssimge_cetveli,SSC);
SSC.kod:='7100';SSC.simge:='BSS';seek(Ssimge_cetveli,filesize(Ssimge_cetveli));Write(Ssimge_cetveli,SSC);
SSC.kod:='F100';SSC.simge:='ÇBS';seek(Ssimge_cetveli,filesize(Ssimge_cetveli));Write(Ssimge_cetveli,SSC);
SSC.kod:='F008';SSC.simge:='ÇBV';seek(Ssimge_cetveli,filesize(Ssimge_cetveli));Write(Ssimge_cetveli,SSC);
SSC.kod:='F004';SSC.simge:='ÇBY';seek(Ssimge_cetveli,filesize(Ssimge_cetveli));Write(Ssimge_cetveli,SSC);
SSC.kod:='8000';SSC.simge:='D';seek(Ssimge_cetveli,filesize(Ssimge_cetveli));Write(Ssimge_cetveli,SSC);
SSC.kod:='7801';SSC.simge:='DÖN';seek(Ssimge_cetveli,filesize(Ssimge_cetveli));Write(Ssimge_cetveli,SSC);
SSC.kod:='5000';SSC.simge:='DÜS';seek(Ssimge_cetveli,filesize(Ssimge_cetveli));Write(Ssimge_cetveli,SSC);
SSC.kod:='7080';SSC.simge:='ESS';seek(Ssimge_cetveli,filesize(Ssimge_cetveli));Write(Ssimge_cetveli,SSC);
SSC.kod:='F200';SSC.simge:='GBS';seek(Ssimge_cetveli,filesize(Ssimge_cetveli));Write(Ssimge_cetveli,SSC);
SSC.kod:='F020';SSC.simge:='GBV';seek(Ssimge_cetveli,filesize(Ssimge_cetveli));Write(Ssimge_cetveli,SSC);
SSC.kod:='F010';SSC.simge:='GBY';seek(Ssimge_cetveli,filesize(Ssimge_cetveli));Write(Ssimge_cetveli,SSC);
SSC.kod:='F080';SSC.simge:='KEV';seek(Ssimge_cetveli,filesize(Ssimge_cetveli));Write(Ssimge_cetveli,SSC);
SSC.kod:='F040';SSC.simge:='KEY';seek(Ssimge_cetveli,filesize(Ssimge_cetveli));Write(Ssimge_cetveli,SSC);
SSC.kod:='F800';SSC.simge:='OKU';seek(Ssimge_cetveli,filesize(Ssimge_cetveli));Write(Ssimge_cetveli,SSC);
SSC.kod:='7002';SSC.simge:='SAD';seek(Ssimge_cetveli,filesize(Ssimge_cetveli));Write(Ssimge_cetveli,SSC);
SSC.kod:='3000';SSC.simge:='SAK';seek(Ssimge_cetveli,filesize(Ssimge_cetveli));Write(Ssimge_cetveli,SSC);
SSC.kod:='4000';SSC.simge:='SAP';seek(Ssimge_cetveli,filesize(Ssimge_cetveli));Write(Ssimge_cetveli,SSC);
SSC.kod:='7040';SSC.simge:='SLB';seek(Ssimge_cetveli,filesize(Ssimge_cetveli));Write(Ssimge_cetveli,SSC);
SSC.kod:='7020';SSC.simge:='SLE';seek(Ssimge_cetveli,filesize(Ssimge_cetveli));Write(Ssimge_cetveli,SSC);
SSC.kod:='7001';SSC.simge:='SOD';seek(Ssimge_cetveli,filesize(Ssimge_cetveli));Write(Ssimge_cetveli,SSC);
SSC.kod:='7010';SSC.simge:='TMB';seek(Ssimge_cetveli,filesize(Ssimge_cetveli));Write(Ssimge_cetveli,SSC);
SSC.kod:='7008';SSC.simge:='TME';seek(Ssimge_cetveli,filesize(Ssimge_cetveli));Write(Ssimge_cetveli,SSC);
SSC.kod:='1000';SSC.simge:='TOP';seek(Ssimge_cetveli,filesize(Ssimge_cetveli));Write(Ssimge_cetveli,SSC);
SSC.kod:='0000';SSC.simge:='VE';seek(Ssimge_cetveli,filesize(Ssimge_cetveli));Write(Ssimge_cetveli,SSC);
SSC.kod:='F400';SSC.simge:='YAZ';seek(Ssimge_cetveli,filesize(Ssimge_cetveli));Write(Ssimge_cetveli,SSC);
SSC.kod:='7820';SSC.simge:='YİA';seek(Ssimge_cetveli,filesize(Ssimge_cetveli));Write(Ssimge_cetveli,SSC);
SSC.kod:='7840';SSC.simge:='YİE';seek(Ssimge_cetveli,filesize(Ssimge_cetveli));Write(Ssimge_cetveli,SSC);
SSC.kod:='7804';SSC.simge:='YDY';seek(Ssimge_cetveli,filesize(Ssimge_cetveli));Write(Ssimge_cetveli,SSC);
SSC.kod:='7802';SSC.simge:='Yİİ';seek(Ssimge_cetveli,filesize(Ssimge_cetveli));Write(Ssimge_cetveli,SSC);
SSC.kod:='2000';SSC.simge:='YÜK';seek(Ssimge_cetveli,filesize(Ssimge_cetveli));Write(Ssimge_cetveli,SSC);
SSC.kod:='7808';SSC.simge:='YVA';seek(Ssimge_cetveli,filesize(Ssimge_cetveli));Write(Ssimge_cetveli,SSC);
SSC.kod:='7810';SSC.simge:='YVK';seek(Ssimge_cetveli,filesize(Ssimge_cetveli));Write(Ssimge_cetveli,SSC);
end;
```

EK-1. (Devam) BB/3 Simülatör Kodları

Birleştirici Kodları

```
procedure TForm4.Birlestir();
  begin
    error:=false;
    sutun:=1;

    Get_nonblank;
    if (satir[sutun]=#0) or (satir[sutun]='/') then
      exit
     else
      begin
        token:=Get_token();

        if token='YER' then begin Yer;editor.cells[1,satir_no]:='';editor.cells[0,satir_no]:='';
        end
       else if yer_sayaci='-1' then
          begin
         mesaj:=Application.MessageBox ('İzlence Yer komutuyla başlamalıdır!','HATA',48);
            hata;
            exit;
          end
       else if token='SON' then editor.cells[1,editor.Row]:='0000'
       else if (token[length(token)]=':') or (token[length(token)]=',') then Yafta
       else deger;
      end;
 end;

 procedure TForm4.Get_nonblank();
 begin
  while (satir[sutun]=' ') and (sutun<length(satir_dizisi[satir_no])) do
    sutun:=sutun+1;
 end;

 function TForm4.Get_token(): string;
```

EK-1. (Devam) BB/3 Simülatör Kodları

```
 begin
    token:='';
    Get_nonblank;
 while not ((satir[sutun]=#0) or (length(satir_dizisi[satir_no])+1=sutun) or (satir[sutun]=' ') or
(satir[sutun]='/')) do
      begin
       token:=token+satir[sutun];
       sutun:=sutun+1;
      end;
      Get_token:=uppercase(token);
  end;

  procedure TForm4.Yer();
  var
  hexa: string[10];
  begin
   token:=Bit_Tamamla(Get_token,3);
   if token<>'' then yer_sayaci:=token;
   if token<>'000' then
    begin
      hexa:=onluga_cevir(token,16);
      if hexa='0' then Hata;
      token:=Get_token;
      if not (token='') then Hata;
    end;
 end;

 procedure TForm4.Yafta();
   var
    i: byte;
   begin

    if yer_sayaci<>'-1' then editor.Cells[0,satir_no]:=yer_sayaci;
  if (((Ord(upcase(token[1]))<65) or (ord(upcase(token[1]))>90))and
(ord(upcase(token[1]))>231) and (ord(upcase(token[1]))>240)and
(ord(upcase(token[1]))>253)and (ord(upcase(token[1]))>105)and
```

EK-1. (Devam) BB/3 Simülatör Kodları

```
(ord(upcase(token[1]))>246) and (ord(upcase(token[1]))>254) and
(ord(upcase(token[1]))>252)) then
    mesaj:=Application.MessageBox ('Yaftanın ilk karakteri harf olmalıdır!','HATA',48)
else if (length(trim(copy(satir_dizisi[satir_no],1,sutun-1))))<>length(token) then
    mesaj:=Application.MessageBox('Yafta tek kelime olmalıdır!','HATA',48)
  else
  begin
    token:=copy(token,1,length(token)-1);
    getdir(0,aktif_dizin);
    aktif_dizin:=aktif_dizin+'\SSC_Dosya.dat';
    AssignFile(Ssimge_cetveli,aktif_dizin);
    if not FileExists(aktif_dizin) then form2.SSimge_Cetveli_Olustur
    else Reset(Ssimge_cetveli);
    for i:=0 to filesize(Ssimge_cetveli)-1 do
     begin
      seek(Ssimge_cetveli,i);
      read(Ssimge_cetveli,SSC);
      if token=SSC.simge then
       begin
        mesaj:=application.MessageBox('Simgeler yafta olarak kullanılamaz!','HATA',48);
        CloseFile(Ssimge_cetveli);
        Hata;
        exit;
       end;
     end;
    CloseFile(Ssimge_cetveli);
    i:=0;
    while i<=50 do
     begin
  if (token=degisken_simge_cetveli[i,0]) and yer_sayaci<>degisken_simge_cetveli[i,1]) then
      begin
       mesaj:=application.MessageBox('Aynı isimli bir yafta tanımlanamaz!','HATA',48);
       hata;
       exit;
      end
else if (token=degisken_simge_cetveli[i,0])and (yer_sayaci=degisken_simge_cetveli[i,1]) then
```

EK-1. (Devam) BB/3 Simülatör Kodları

```
        begin
          token:=Get_token();
          if token='SON' then
             begin
              editor.cells[1,editor.Row]:='0000';
              break;
             end;
          Deger;
          break;
         end
        else if degisken_simge_cetveli[i,0]='' then
         begin
          degisken_simge_cetveli[i,0]:=token;
          degisken_simge_cetveli[i,1]:=yer_sayaci;
          token:=Get_token();
          if token='SON' then
             begin
               editor.cells[1,editor.Row]:='0000';
               break;
             end;
          Deger;
          break;
         end;
        i:=i+1;
        end;
   end;
end;

procedure TForm4.Deger();
var
  i: byte;
  Muhteva, Hatira: string[5];
  simge: Boolean;
begin

 if yer_sayaci<>'-1' then editor.Cells[0,satir_no]:=yer_sayaci;
```

EK-1. (Devam) BB/3 Simülatör Kodları

```
if token='' then
 yer_sayaci:=Bit_Tamamla (onluktan_cevir((strtoint(onluga_cevir(yer_sayaci,16))-1),16),3);
  while token<>'' do
   begin
    simge:=false;
    Hatira:='-1';
    if (token[1]='=') then
      begin
        token:=copy(token,2,length(token)-1);
        Hatira:=Sabit;
      end
    else
      begin
       getdir(0,aktif_dizin);
       aktif_dizin:=aktif_dizin+'\SSC_Dosya.dat';
       AssignFile(Ssimge_cetveli,aktif_dizin);
       if not FileExists(aktif_dizin) then form2.SSimge_Cetveli_Olustur
       else Reset(Ssimge_cetveli);
       for i:=0 to filesize(Ssimge_cetveli)-1 do
          begin
            seek(Ssimge_cetveli,i);
            read(Ssimge_cetveli,SSC);
            if token=SSC.simge then
              begin
                Hatira:=SSC.kod;
                simge:=true;
                break;
              end;
          end;
       if i=filesize(Ssimge_cetveli) then
          begin
            CloseFile(Ssimge_cetveli);
            i:=0;
            while (i<=50) and (degisken_simge_cetveli[i,0]<>'') do
              begin
                if token=degisken_simge_cetveli[i,0] then
```

EK-1. (Devam) BB/3 Simülatör Kodları

```
                    begin
                      Hatira:=degisken_simge_cetveli[i,1];
                      break;
                    end;
                   i:=i+1;
                  end;
               end;
          end;
          if simge=true then  CloseFile(Ssimge_cetveli);
           if Hatira<>'-1' then
            begin
             Hatira:=Bit_Tamamla(Hatira,4);
              Muhteva:=Bit_Tamamla(onluktan_cevir((strtoint(onluga_cevir(Muhteva,16)) or
              strtoint(onluga_cevir(Hatira,16))),16),4);
            end
           else Hata;
           token:=Get_token;
       end;
     if error=false then
        begin
          if Muhteva='' then editor.Cells[1,satir_no]:='0000'
          else editor.Cells[1,satir_no]:=Muhteva;
        end;
   yer_sayaci:=Bit_Tamamla(onluktan_cevir((strtoint(onluga_cevir(yer_sayaci,16))+1),16),3);
   end;

function TForm4.Sabit():string;
  var
   deger: string[5];
  begin
if (token[1]='-') then deger:=onluktan_cevir((strtoint(onluga_cevir('FFFF',16))-
strtoint(onluga_cevir(token,16))+1),16)
    else Deger:=token;
    if length(deger)>4 then Hata
    else  Sabit:=Deger;
end;
```

EK-1. (Devam) BB/3 Simülatör Kodları

```
procedure Tform4.Satiri_isle();
 var
  i,j: integer;
begin
 satir_dizisi[editor.Row]:=editor.Cells[editor.Col,editor.Row];
 satir:=editor.Cells[editor.Col,editor.Row];
 satir_no:=editor.Row;
 if editor.Cells[0,editor.Row]<>'' then yer_sayaci:=editor.Cells[0,editor.Row];
 birlestir();
 if yer_sayaci<>'-1' then
     begin
      Birinci_derle;
      ikinci_derle;
     end;
end;

procedure TForm4.Birinci_derle();
var
 i: byte;
begin
 satir_no:=1;
 yer_sayaci:='-1';
 for i:=0 to 50 do
   begin
     if degisken_simge_cetveli[i,0]<>'' then
      begin
        degisken_simge_cetveli[i,0]:='';
        degisken_simge_cetveli[i,1]:='';
      end
     else break;
   end;

  for satir_no:= 1 to editor.RowCount-1 do
      begin
         satir_dizisi[satir_no]:=editor.Cells[editor.Col,satir_no];
         satir:=editor.Cells[editor.Col,satir_no];
```

EK-1. (Devam) BB/3 Simülatör Kodları

```
            birlestir()
        end;
    end;

    procedure TForm4.ikinci_derle();
    begin
       for satir_no:= 1 to editor.RowCount-1 do
        begin
            if editor.Fonts[0,satir_no].Color=clRed then Hata_Sil;
            satir:=editor.Cells[editor.Col,satir_no];
            birlestir();
        end;
          satir_ilerlet;
    end;
```

EK-2. BB/3 Simülatörü Kullanım Kılavuzu

BB/3 SİMÜLATÖRÜ VE BİRLEŞTİRİCİ

Delphi 7.0 ile oluşturulmuş olan BB/3 donanım simülatörü 4 pencereden oluşmaktadır. Bu pencereler;

- BB/3 simülatörü,
- Birleştirici,
- Sabit simge cetveli
- Sayı çeviricidir.

BB/3 Simülatörü

BB/3'ün işleyişinin mantıksal tasarımını irdelemek için, "donanım tasarım dili"ndeki anlatımı Delphi 7.0 diline çevrilerek bir donanım simülatörü oluşturulmuştur. Bu simülatörü geliştirmek isteyecekler açısından yazılımın sistematik olmasına dikkat edilmiştir. Bu sayede Donanım Tasarımı dersini alan öğrenciler simülatör kodlarını inceleyip anlayabilirler ve hatta donanım üzerinde yapmak istedikleri değişiklikleri simülatöre uyarlayıp, simülatörü değiştirebilirler.

BB/3 simülatörünün 4K büyüklüğünde (her kelimesi 16 bitlik) ana hafızası vardır. BB/3 simülatörünün ana hafızası; Const Azami_adr=4095; HAFIZA: Array "[0..Azami_adr] of string[4]" şeklinde tanımlanmıştır. Ana hafızayı temsil etmek için bir tablo kullanılmıştır.

Simülatörün ana penceresinde bulunan bu tablo içerisinde 4K'lık hafıza adresleri ve bu hafıza adreslerinin verileri gösterilmektedir. BBSD dili ile yazılmış kaynak kodun birleştirilmesiyle oluşturulan amaç kodların değerleri yine bu tablo içerisinde gösterilmektedir. Yığıtı daha kolay incelemek için birinci tablonun altında "FF0" dan "FFF" hafıza adresini gösteren ikinci bir tablo bulunmaktadır.

Ayrıca simülatörün ana ekranında, giriş işlemlerinin yapılabilmesi için klavye, çıkış işlemlerinin sonuçlarının görülebilmesi için bir ekran bulunmaktadır. BB/3 donanım

EK-2. (Devam) BB/3 Simülatörü Kullanım Kılavuzu

simülatöründe 16 bitlik kayıtlıklar word, 8 bitlik kayıtlıklar byte, bayraklar ise bit olarak tanımlanmıştır. Bir kayıtlığın denetim girişlerinden birisinin etkin olmasının sonucu aynı anda yani "0" sürede çıkıştan almak mümkün değildir. Bu nedenle her kayıtlığın giriş ve çıkışını temsil etmek için, kayıtlıkların girişi ve çıkışı ayrı ayrı tanımlanmıştır.
Örneğin; BI_G (Birikeç kayıtlığının girişi), BI_C (Birikeç kayıtlığının çıkışı).

Simülatör ana ekranında BB/3 kayıtlıklarının, bayraklarının ve ikililerinin değerlerini gösterir yazı kutucukları bulunmaktadır. Bu yazı kutucuklarından Kİ, Yİ, E ve Bİ değerleri kullanıcı tarafından istenildiğinde değiştirilebilir. BB/3 izlencelerini zaman-zaman, devir-devir, komut-komut, serbest veya duraklı serbest olarak çalıştırabilmek için işlem düğmeleri bulunmaktadır. Bu sayede seçime bağlı olarak, her "zaman adımı", devir veya komut bitiminde Kİ, Yİ, HAK, HDK, KK, d, z, E, Bİ, GİRK, GİRB, GBİ, ÇIKK, ÇIKB, ÇBİ, KYV ve Ç kayıtlık ve bitlerinin alacağı değerler, ana hafızada ve çıkış ekranında olan değişiklikler aynı pencere içersinde, kullanıcıya gösterilir.

Ayrıca kullanıcının daha kolay takip edebilmesi için, Kİ ve Yİ'nin işaret ettiği hafıza hücreleri renklendirilmiştir. Kİ'nin işaret ettiği hafıza hücresi sarı renkle, Yİ'nin işaret ettiği hafıza hücresi ise açık mavi renkle gösterilmiştir. Resim 2.'de BB/3 donanım simülatörünün ana ekranı gösterilmektedir. BB/3 simülatörü BB/1 simülatörünün tüm özelliklerine sahiptir. BB/1 simülatöründen farklı olarak giriş-çıkış ve yığıt işlemlerinin simülasyonu da yapılmaktadır.

Simülatör çalıştığında kayıtlıklara, bayraklara ve bitlere BB/3 bilgisayar modeline göre, Resim 2.'de görünen ilk değerleri aktarılır. Yığıt işaretçisi ilk başta yığıtın en dibi olan "FFF" değerini gösterir. Herhangi bir veri girişi olmadığından ana hafızanın tüm kelimelerine "0000" değeri atanır.

EK-2. (Devam) BB/3 Simülatörü Kullanım Kılavuzu

Herhangi bir BB/3 izlencesinin yürütülebilmesi için öncelikle izlencenin makine kodlarının (Amaç kodlarının) BB/3 simülatörü ana hafızasına yüklü olması gerekir. Bunun için makine kodları BB/3 simülatörü ana hafıza kelimelerine girilebilir veya daha önce amaç kodları şeklinde kaydedilmiş olan izlenceler "Aç" penceresinden seçilerek yüklenebilir. Üçüncü bir yöntem olarak birleştirici kullanılarak simgesel kodlar kullanılarak yazılmış izlencelerin amaç kodları BB/3 simülatörü ana hafızasına aktarılabilir. BB/3 simülatörü ana hafızasına yüklenen izlence kodları "zaman_zaman_calistir" isimli bir altyordam vasıtasıyla, BB/3 izlencesinin komut işaretçisinde belirtilen adresten itibaren yürütülmesine başlanmaktadır. BB/3 simülatörü işlemcisinin çalışır durumda olup olmadığını çalış biti (C) ile belirliyoruz. C=1 olduğu sürece "zaman_zaman_calistir" altyordamı çalışmaya devam edebilmektedir. C=0 olduğunda BB/3 işlemcisi duracağından, ana hafızasında bulunan izlencelerin yürütülmesi de durur.

"zaman_zaman_calistir" altyordamı çalıştığında ilk başta Kayıtlıkların çıkış değerleri, bayrakların, bitlerin ve devir takip ikilisinin (A, B) değerleri aktarılır. Daha sonra kod çözücüsü değerleri aktarılır:

```
For i:=0 To 7 do K[i]:=0;
K[(KK and $7000) SHR 12]:=1;              {İşlem Kodu Kod Çözücü}
For i:=15 downto 0 do  V[i]:=(KK SHR i) and $0001; {KK nin  bit değerleri}

For i:=0 To 3 do Begin D[i]:=0;Z[i]:=0; End;
   D[2*B+A]:=1;      {BA Kod Çözücü}
   Z[SS]:=1;         {SS Kod Çözücü}
```

Kod çözücülerin önceki değerleri "0"landıktan sonra, İşlem Kodu kod çözücüsünün çıkış değeri "1"lenir. K[7]=1 olursa ifa edilecek komutun kayıtlık, giriş çıkış veya yığıt komutu olduğunu belirtir. K[7]'den başka bir çıkış "1"lenirse ifa edilecek komutun hafıza adresleyen komut olduğunu belirtir. Daha sonra "B" ve "A" devir

EK-2. (Devam) BB/3 Simülatörü Kullanım Kılavuzu

takip ikilisine göre Devir belirlenir. D[0]:=1 değeri getir devrini, D[1]:=1 değeri dolaylı devri, D[2]:=1 değeri ifa devrini, D[3]:=1 değeri ise kesinti devrini belirtir. Devir belirlendikten sonra SS kod çözücüsü (Sıra sayacı) değerine göre zaman belirlenir. Daha sonra KK'nın uygun bitleri "1"lenir. Kod çözücü çıkışlara uygun devir ve zamandaki kayıtlık aktarım dili satırları taranır ve şartları geçerli olan mikroişlemler gerçekleştirilir. Komut işaretçisi olan Kİ'nin bir zaman adımı içinde birden fazla ilerletilmesi hatasını engellemek için KI_SAY biti kullanılır. Herhangi bir mikroişlem sonucu KI_SAY bit değeri "1" olmuş ise Kİ ilerletilir. Kayıtlıkların girişine gelen değerler çıkışa aktarılır. Kayıtlık, bayrak, bit ve ikili değerleri ekranda sergilenir. Sıradaki zamana geçilir. BB/3 simülatörünün ana hafıza kelimelerinin muhtevaları güncellenir.

"Zaman_zaman_calistir" altyordamının genel mantığı aşağıdaki gibidir:

1. Bit, bayrak, ikililer ve kayıtlık çıkışlarının değerlerini aktar.
2. Simüle edilecek devire ve zamana göre kod çözücü çıkış değerlerini belirle.
3. Kayıtlık aktarım dili satırlarını tara, şartları geçerli olan mikroişlemleri gerçekleştir.
4. Kayıtlık girişlerine gelen değerleri, kayıtlık çıkışlarına aktar.
5. Bit, bayrak, ikililer, kayıtlık değerleri, BB/3 simülatörü ana hafızası ve eğer varsa çıkış aygıtındaki değerleri sergile.
6. Sıradaki zamana geç.
7. İşleyiş devam edecek ise 2 no.lu satıra git, yoksa çık.

Yığıt işaretçisi, yığıtın dibi olan "FFF" değerini gösteriyorken "DÖN" komutu kullanıldığında yığıt işaretçisinin değeri "1000" olur. Dön komutunun son zamanında Yİ'nin bit11 değeri kontrol edilir. Eğer Yİ'nin bit11 = 0 ise çalış biti (C=0) sıfır yapılarak izlencenin bittiğini gösterir.

BB/3 simülatörü giriş birimi olan klavyeden bir tuşa basıldığında, tuşun şekilcik kodu Giriş kayıtlığına (GİRK) aktarılır. Giriş donanımı, GİRK'ya yeni bir şekilcik

EK-2. (Devam) BB/3 Simülatörü Kullanım Kılavuzu

kod değeri geldiğini belirtmek için GİRB'i "1" yapar. OKU komut ile GİRK'daki şekilcik kodu birikece aktarılabilir. OKU komutu, GİRK'daki şekilcik kodunu birikece aktarır ve GİRK'da taze kod olmadığını göstermek için GİRB'yi "0" yapar. Eğer yazıcı müsait ise (ÇIKB=1) YAZ komutu ile birikece alınan şekilcik kodunu ÇIKK'ya aktarır. YAZ komutu aynı zamanda yazıcının meşgul durumuna geçtiğini belirtmek için ÇIKB'yi "0" yapar. Yazıcı yazma işlemini bitirdikten sonra tekrar müsait duruma geçtiğini göstermek için ÇIKB'yi "1" yapar. BB/3 simülatörü gerçeğe uygun bir simülatördür. Yazıcının donanım gecikmesini devreye sokmak için YAZ komutu ifa edilirken, Print_suresi isimli bir değişkenin değeri "100" yapılır. "Print_sureci" isimli bir altyordam yardımıyla, Print_süresi değişkeninin değeri sıfır oluncaya kadar her zamanın sonunda birer azaltılır. Değişkenin değeri sıfır olduğunda, kodu ÇIKK'da olan şekilcik, BB/3 simülatör penceresindeki çıkış birimine yazılır.

Birleştirici

BBSD dili kullanılarak BB/3 bilgisayar modelinde simgesel kod kullanılarak gerçek ve kapsamlı izlenceler yazılabilmesi için, BB/3 donanım simülatöründe bir birleştirici de mevcuttur. Bu birleştirici ciddi izlence örnekleri yazmada ve onları okumada kolaylık sağlar. Bu birleştirici Sabit simgeler (BB/3 komutları), tanımlanmış simgeler (Yaftalar), sabit değerler, değer ifadeleri, makrolar ve yardımcı komutlar kullanılarak, kaynak kodları girilen izlencelerin amaç kodlarını oluşturarak, BB/3 donanım simülatörünün ana hafızasına aktarmayı sağlar.

Birleştirici programı çalıştırıldığında ekrana gelen pencerede, izlencelerin yazılabilmesi için gerekli bir tablo bulunmaktadır. Tablo üç sütundan oluşmaktadır. BBSD dili şeklindeki izlence satırları en sağda bulunan sütuna yazılır. BBSD dili ile yazılmış olan izlencede hata yoksa birleştirici her satıra tekabül eden amaç kodunu "MUHTEVA" sütununa aktarır. BBSD dili ile yazılan izlence satırlarının ana

hafızada yerleşecekleri yerleri “YER” sütununda belirtilir. Yardımcı komutlardan oluşan satırın “MUHTEVASI” ve “YER” değeri belirtilmez.

Sabit Simge Cetveli

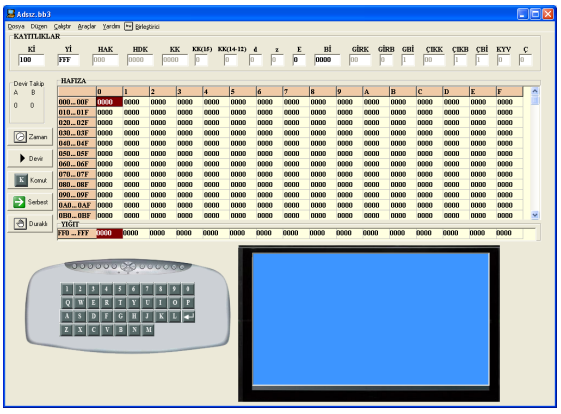
BB/3 bilgisayar modelinin komutlarının listelendiği, Sabit simge cetveli penceresi açılır. Bu pencerede BB/3 bilgisayar modeli komutlarının simgeleri ve değerleri alfabetik sırayla, bir tabloda gösterilir.

Sayı çevirici

Sayı çevirici yardımcı programı değeri girilen ve tabanı seçilen bir sayının istenilen tabandaki karşılığını bulmayı sağlar. Taban olarak “2, 4, 8, 10, 16” değerleri seçilir.

BB/3 DONANIM SİMÜLATÖRÜNÜN ÇALIŞTIRILMASI

Simülatör programını çalıştırmak için BB3 simgesine tıklanması yeterlidir. Programın ana penceresi Resim 2.1’de gözükmektedir.



Resim 2.1. BB/3 donanım simülatörünün ana penceresi.

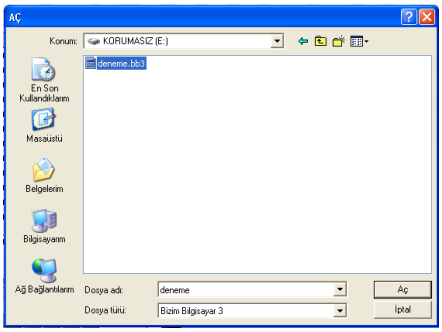
Dosya Menüsü

Dosya menüsünde Aç, Kaydet, Farklı Kaydet ve Çıkış seçenekleri vardır.

EK-2. (Devam) BB/3 Simülatörü Kullanım Kılavuzu

Aç

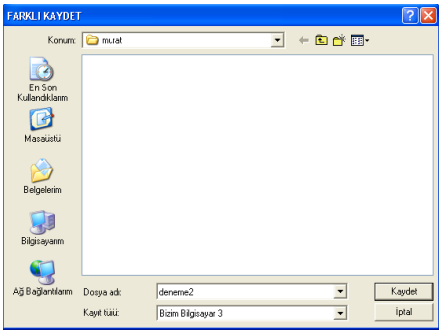
Aç komutu seçildiğinde “AÇ” penceresi açılır. Bu pencere; daha önce kayıtlı olan, makine kodları ile yazılmış BB/3 (bb3 uzantılı) izlencelerini, BB/3 simülatöründeki ana hafızaya aktarmaya sağlar.



Resim 2.2. Aç komutu.

Kaydet

Kaydet komutu seçildiğinde “KAYDET” penceresi açılır. Bu pencere, BB/3 simülatöründeki ana hafızada bulunan makine dili kodlarını bir dosyaya kaydetmek için kullanılır. Dosyanın uzantısı “bb3” olur. Eğer kaydedilecek dosya, daha önce kayıtlı ise dosyanın içeriği güncellenir. Dosya daha önce kayıtlı değilse, ilk defa kaydediliyorsa “FARKLI KAYDET” penceresi açılır.



Resim 2.3. Farklı Kaydet Penceresi

EK-2. (Devam) BB/3 Simülatörü Kullanım Kılavuzu

Farklı Kaydet

Farklı Kaydet komutu seçildiğinde "FARKLI KAYDET" penceresi açılır. Bu pencere, BB/3 simülatöründeki ana hafızada bulunan makine dili kodlarını farklı bir yerdeki farklı bir dosyaya kaydetmek için kullanılır. Dosyanın uzantısı "bb3" olur. Eğer kaydedilecek dosya, daha önce kayıtlı ise dosyanın içeriği güncellenir.

Çıkış

Çıkış komutu seçildiğinde BB/3 simülatörü kapatılır.

Düzen menüsü

Düzen menüsünde Kes, Kopyala, Yapıştır, Sil ve Tümünü Sil seçenekleri vardır. Bu seçenekler yardımıyla ana hafızanın gösterildiği tablo içerisinde seçilen işlem yapılır. Hafıza hücrelerini seçmek için seçilecek hücre üzerinde farenin sol tuşuna tıklayıp fare sürüklenmelidir. Düzen menüsünde bulunan seçenekler, BB3 simülatörü ana hafızasının gösterildiği tablo üzerinde farenin sağ tuşuna tıklandığında da ortaya

çıkar. Yapıştır seçeneği normalde aktif değildir, seçeneklerde görülmez. Yapıştır seçeneğinin aktif olması için daha önce kesme veya kopyalama işleminin yapılması gereklidir.

Resim 2.4. Düzen menüsü

EK-2. (Devam) BB/3 Simülatörü Kullanım Kılavuzu

Kes

Kes komutu, seçimi yapılan BB/3 simülatörü ana hafıza kelime veya kelimelerinin muhtevalarını kesip bilgisayarın hafızasına alır.

Kopyala

Kopyala komutu, seçimi yapılan BB/3 simülatörü ana hafıza kelime veya kelimelerinin muhtevalarını kopyalarını bilgisayarın hafızasına alır.

Yapıştır

Yapıştır komutu, Kes veya Kopyala ile bilgisayar hafızasına aktarılan ana hafıza kelimelerinin muhtevalarını, aktif olan ana hafıza kelimesinden itibaren yapıştırır.

Sil

Sil komutu, seçimi yapılan BB/3 simülatörü ana hafıza kelime veya kelimelerinin muhtevalarını siler, bu hafıza kelimelerine "0000" değeri aktarır.

Tümünü Sil

Tümünü Sil komutu, BB/3 simülatörü ana hafıza kelimelerinin tümünün muhtevalarını siler ve tüm kelimelere "0000" değeri aktarır. Ayrıca tüm bit, bayrak ve kayıtlık değerlerini de silerek ilk değerlerine kurar.

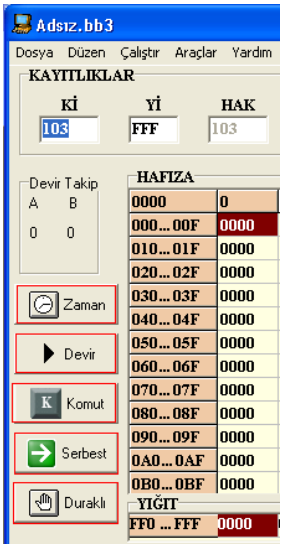
Çalıştır Menüsü

Çalıştır menüsü BB/3 simülatörünün ana hafızasında yazılı olan izlence örneklerinin yürütülüş tarzını belirler. BB/3 simülatöründe beş izleniş tarzı vardır:

EK-2. (Devam) BB/3 Simülatörü Kullanım Kılavuzu

Her zaman adımı bitiminde (Zaman-Zaman), her devir bitiminde (Devir-Devir), her komutun ifa devri bitiminde (Komut-Komut), serbest (izleniş yok) ve duraklı serbest(Durak noktalarında izleniş var). Resim 2.5 – (b)'de gözüktüğü gibi bu izlence yürütülüş tarzları BB/3 simülatörü ana penceresindeki komut düğmeleri yardımıyla da seçilebilir.

Resim 2.5. (a). Çalıştır Menüsü



Resim 2.5 (b). Ana penceredeki komut düğmeleri

Zaman-Zaman

Zaman-Zaman çalıştır komutu seçildiğinde, izlence bir zaman adımı çalışır, yani zaman_zaman_calistir altyordamı bir defa çağrılır.

EK-2. (Devam) BB/3 Simülatörü Kullanım Kılavuzu

Devir-Devir

Devir-Devir çalıştır komutu seçildiğinde, izlence bir devir çalışır, yani z[3]=1 olana kadar, zaman_zaman_calistir altyordamı çağrılır.

Komut-Komut

Komut-Komut çalıştır komutu seçildiğinde, izlence bir komut ifa edilene kadar çalışır, yani d[2]=1 ve z[3]=1 olana kadar, zaman_zaman_calistir altyordamı çağrılır.

Serbest

Serbest çalıştır komutu seçildiğinde, izlence otomatik olarak çalışır. İzlence bitimine kadar zaman_zaman_calistir altyordamı çağrılır.

Duraklı

Duraklı Serbest çalıştır komutu seçildiğinde, izlence otomatik olarak çalışır. İzlence bitimine veya Ki'nin değeri bir durak noktasına eşit olana kadar zaman_zaman_calistir altyordamı çağrılır. Özel olarak incelenmek istenen hafıza kelimelerine durak noktası eklenir. Durak noktası eklemek istenilen hafıza kelimesinin üzerinde farenin sağ tuşuna tıklanır. Çıkan menüden "Durak Noktası Ekle" seçilir. Durak noktası eklenen hafıza kelimesinin rengi kırmızı olur. Durak noktası ekli hafıza kelimesinin durak noktasını kaldırmak için fare hafiza kelimesinin üzerinde iken sağ tuş tıklanır. Çıkan menüden "Durak Noktasını Kaldır" seçilir.

Araçlar Menüsü

Araçlar menüsü BB/3 simülatörünü kullanırken yararlanılabilecek araçlardan oluşmaktadır. Araçlar menüsünde, Sayı Çevirici ve Sabit Simge Cetveli vardır.

EK-2. (Devam) BB/3 Simülatörü Kullanım Kılavuzu

Resim 2.6. Araçlar menüsü

Sayı Çevirici

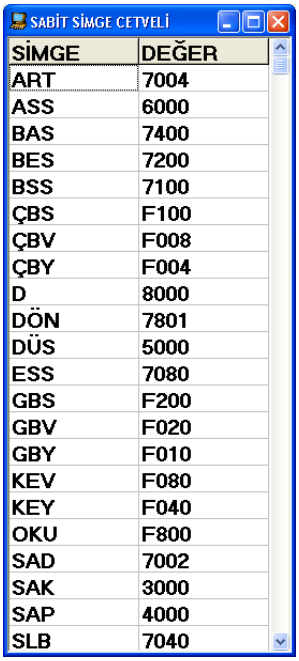
Sayı Çevirici komutu seçildiğinde, Sayı çevirici yardımcı program çalışır. Resim 2.7'de görülen bu program; değeri girilen ve tabanı seçilen bir sayının istenilen tabandaki karşılığını bulmayı sağlar. Taban olarak "2, 4, 8, 10, 16" değerleri seçilebilir.

Resim 2.7. Sayı çevirici yardımcı programı.

Sabit Simge Cetveli

Sabit Simge Cetveli seçeneği seçildiğinde, BB/3 bilgisayar modelinin komutlarının listelendiği, Sabit simge cetveli penceresi açılır. Resim 2.8'de görülen bu pencerede BB/3 bilgisayar modeli komutlarının simgeleri ve değerleri alfabetik sırayla, bir tabloda gösterilir.

EK-2. (Devam) BB/3 Simülatörü Kullanım Kılavuzu



Resim 2.8. Sabit simge cetveli penceresi

Yardım Menüsü

Yardım menüsündeki Hakkında komutu; BB/3 simülatörü ve menüleri ile ilgili bilgi sahibi olunmayı sağlayan kullanıcı kılavuzuna ulaşmayı sağlar.

BİRLEŞTİRİCİ

BBSD dili kullanılarak BB/3 bilgisayar modelinde simgesel kod kullanılarak gerçek ve kapsamlı izlenceler yazılabilmesi için, BB/3 donanım simülatöründe bir birleştirici de mevcuttur. Bu birleştirici ciddi izlence örnekleri yazmada ve onları okumada kolaylık sağlar. Bu birleştirici Sabit simgeler (BB/3 komutları), tanımlanmış simgeler (Yaftalar), sabit değerler, değer ifadeleri, makrolar ve yardımcı komutlar kullanılarak, kaynak kodları girilen izlencelerin amaç kodlarını oluşturarak, BB/3 donanım simülatörünün ana hafızasına aktarmayı sağlar.

EK-2. (Devam) BB/3 Simülatörü Kullanım Kılavuzu

Birleştirici programı çalıştırıldığında ekrana gelen pencerede, izlencelerin yazılabilmesi için gerekli bir tablo bulunmaktadır. Tablo üç sütundan oluşmaktadır. BBSD dili şeklindeki izlence satırları en sağda bulunan sütuna yazılır. BBSD dili ile yazılmış olan izlencede hata yoksa birleştirici her satıra tekabül eden amaç kodunu "MUHTEVA" sütununa aktarır. BBSD dili ile yazılan izlence satırlarının ana hafızada yerleşecekleri yerleri "YER" sütununda belirtilir. Yardımcı komutlardan oluşan satırın "MUHTEVASI" ve "YER" değeri belirtilmez.

Birleştiricinin çalışma prensipleri

Birleştirici BBSD'nin tanımına uygun olarak çalışmaktadır. BBSD dilinde birleştiriciye yazılan izlence, "YER" yardımcı komutuyla başlamalıdır. Bu sayede izlencenin, 4K olan ana hafızada yerleştirilmeye başlanacağı yer belirlenmiş olur. İzlencede birleştirilecek olan her satırın yer değeri vardır. Herhangi bir satırın YER değerini belirlemek için BBSD dilinin "YER" yardımcı komutu kullanılır. YER komutu kullanılmadığında, satırın yer değeri, bir önceki satırın yer değerinin bir fazlasıdır. YER değeri olarak "000" ile "FFF" arasında onaltılık tabanda bir sayı girilmelidir. Aksi halde hatalı bir giriş yapılmış olur. Sadece yardımcı komutlardan oluşan bir satırın "YER" değeri belirlenmez.

Birleştirici programında "/" yardımcı komutu yorum eklemek için kullanılır. "/" işaretinden sonra gelen ifadeler birleştirici tarafından değerlendirilmeye alınmaz. Birleştirici izlenceyi "SON" yardımcı komutu satırına kadar derler. "SON" yardımcı komutundan sonraki satırlar birleştirme işlemine tabi tutulmaz.

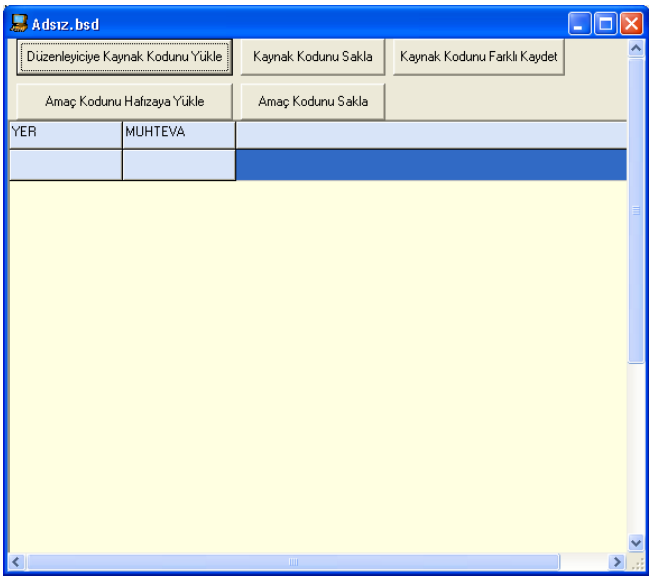
İzlencelerin makine diline çevrilişleri iki safhada olur. Birinci safhada yaftaların adres değerleri belirlenerek "Değişken Simge Cetveli" oluşturulur. Değişken simge cetveli, izlencede kullanılan yaftaların isimleri ve işaret ettikleri adres değerlerini barındırır. Cetvel, yafta isimlerine göre alfabetik sırada oluşturulur. Simgesel kodla yazılan izlencelerde kullanılan yafta simgeleri bir harf ile başlayıp "," veya ":" işareti ile sonlanmalıdır. Bir yafta tek kelimeden oluşmalıdır ve herhangi bir satırda

tanımlanan yafta simgesi başka bir satırda tekrar kullanılamaz. Aksi halde hata meydana gelir. Bir satırda sadece yafta tanımlanmış ise YER değeri artırılmaz. İkinci safhada ise birleştirme işi yapılır ve eğer hata yok ise, her satıra tekabül eden amaç kodu ve yer değeri oluşturulur.

Birleştirici penceresinde beş farklı işlem yapılabilmektedir. Bu işlemlere komut düğmeleri yardımıyla ulaşılabilir. Bunlar; "Düzenleyiciye Kaynak Kodunu Yükle", "Kaynak Kodunu Sakla", "Kaynak Kodunu Farklı Kaydet", "Amaç Kodunu Hafızaya Yükle" ve "Amaç Kodunu Sakla" işlemleridir.

BİRLEŞTİRİCİNİN ÇALIŞTIRILMASI

Birleştirici komutuna seçildiğinde birleştirici penceresi ekrana gelir. Birleştirici penceresinde beş farklı işlem yapılabilmektedir. Bu işlemlere komut düğmeleri yardımıyla ulaşılabilir. Bunlar; "Düzenleyiciye Kaynak Kodunu Yükle", "Kaynak Kodunu Sakla", "Kaynak Kodunu Farklı Kaydet", "Amaç Kodunu Hafızaya Yükle" ve "Amaç Kodunu Sakla" işlemleridir.



Resim 2.9. Birleştirici Penceresi

EK-2. (Devam) BB/3 Simülatörü Kullanım Kılavuzu

Düzenleyiciye kaynak kodunu yükle

Düzenleyiciye Kaynak Kodunu Yükle komutu, BBSD ile yazılıp daha önce bilgisayara kaydedilmiş olan izlencelerin seçilip, bu izlenceleri birleştiriciye aktarılmasını sağlar. Simgesel dil dosyalarının uzantıları "bsd" dir.

Kaynak kodunu sakla

Kaynak Kodunu Sakla komutu, Birleştirici penceresinde simgesel kodları yazılmış olan izlenceyi "bsd" uzantılı olarak bilgisayara kaydetmeyi sağlar.

Kaynak kodunu farklı kaydet

Kaynak Kodunu Farklı Kaydet komutu, Birleştirici penceresinde simgesel kodları yazılmış olan izlenceyi "bsd" uzantılı olarak farklı bir konuma farklı bir isimle kaydetmeyi sağlar.

Amaç kodunu hafızaya yükle

Amaç Kodunu Hafızaya Yükle komutu, Birleştirici penceresinde simgesel kodları yazılmış olan izlencenin amaç kodlarını BB/3 simülatörünün ana hafızasına yüklemeyi sağlar. Eğer izlence satırının herhangi birinde hata varsa amaç kodu simülatörün ana hafızasına aktarılamaz. İzlencede herhangi bir hata yoksa, her satırın muhtevası, yer sütunundaki adrese eşit ana hafıza kelimesine aktarılır.

Amaç kodunu sakla

Amaç Kodunu Sakla komutu, Birleştirici penceresinde simgesel kodları yazılmış olan izlencenin amaç kodlarını "bb3" uzantılı bir dosya olarak bilgisayara kaydetmeyi sağlar.

EK-2. (Devam) BB/3 Simülatörü Kullanım Kılavuzu

BİZİM BİLGİSAYAR SİMGESEL DİLİ (BBSD)

Simgesel programlar, makine dili programlar gibi alt alta yazılan simgesel komutlardan oluşur. İzlencelerin makine dili şeklinde olan kodlarına "Amaç kodu", simgesel komutlar şeklinde olan kodlarına "Kaynak kodu" demekteyiz. Simgesel komutlar, makine dili karşılığı olanlar ve olmayanlar diye iki cins olur. Makine dili olmayan simgesel komutlara *yardımcı komutlar* denir. Yardımcı komutlar birleştiriciye hitap ederler.

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.

Çizelge 2.1 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

2. *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.

EK-2. (Devam) BB/3 Simülatörü Kullanım Kılavuzu

3. *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.

4. *Makro*: Herhangi bir satırın solunda bir harfle başlayan, iki defa eşittir (==) işareti ile biten ve arada boşluk bulunmayan bir harf rakam dizgisi makro olarak tanımlanmıştır. Makronun muhtevası ise iki eşittir işaretinin sağında bulunan komutların muhtevalarının VEYA'lanmasıyla oluşan değere eşittir.

5. *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.

BBSD'nin tanımı, bir sayfaya sığacak kısalıkta olmakla birlikte, bir simgesel dilde olması gereken tüm özellikler BBSD'nde mevcuttur.

<u>BBSD'nin tanımı</u>:

1. Her satırda en çok bir hafıza yeri muhtevasının (komut veya veri) ifadesi olabilir.

2. Makine komutlarının hafızada denk geleceği yerin tayini için, sıradaki makine komutlarının hangi yerden itibaren dizilmesi istendiği, bir YER yardımcı komutuyla belirtilir. Ör. YER 200

3. Makine komutlarının sayısal kodları yerine simgeleri yazılabilir.

4. 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.

EK-2. (Devam) BB/3 Simülatörü Kullanım Kılavuzu

5. 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.

6. 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. 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.
   Örnek: K1: ART

   SAP K1

7. Simgesel programa açıklayıcı ifadeler ilave edebilmek için "/" işareti kullanılır. Bu işaretin sağında kalan şekilcikler makine diline çevrilişte dikkate alınmaz.

8. BBSD'de izlence yazarken birden çok komuttan oluşan bir satır, izlencede birden çok tekrarlanıyorsa bu satıra tekabül eden bir makro oluşturulabilir. Makroya bir isim verilir daha sonra yan yana iki defa eşittir (=) işareti konulur ve makroya tekabül eden komutlar yazılır ve makronun muhtevası oluşur. Makro izlencenin başka yerlerinde çağrıldığında çağrılan yere makronun muhtevası atanır.
   Örneğin; TAMLA == TMB ART olarak tanımlanan TAMLA makrosunun muhtevası "7014" olur. İzlencenin başka bir yerinde TAMLA denildiğinde oraya "7014" muhtevası atanır.

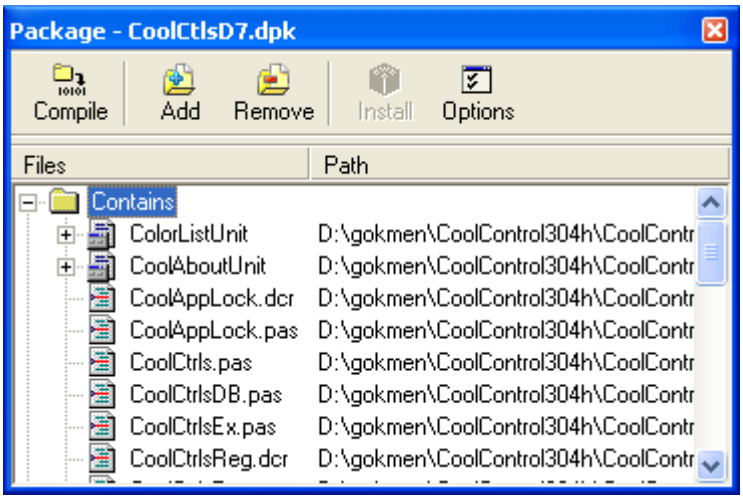
EK-2. (Devam) BB/3 Simülatörü Kullanım Kılavuzu

9. Hafıza adresleyen komutlarda hatıradan önce bulunan bir D harfi, indisli adresleyişin olduğunu belirtir.

10. Simgesel programın sonunu belirtmek için SON yardımcı komutu kullanılır.

EK-3. Coolcontrol paketinin Delphi 7.0 Programlama diline yüklenmesi

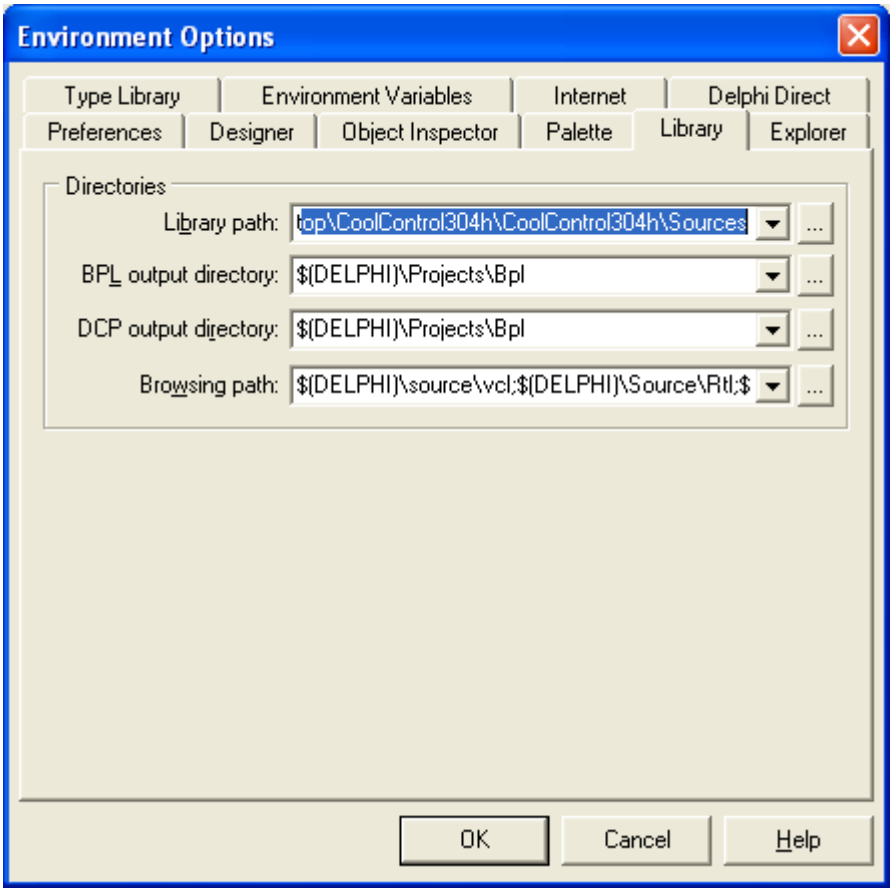
Coolcontrol paketinin kurulumuna ait dosyaların bulunduğu Coolcontrol klasörü bilgisayara kopyalanır. Bu klasörün içerisinde olan Source klasörünün içerisinde bulunan Delphi7 klasörünün içerisinde bulunan dosyalar bir üst düzeyde olan Source klasörüne kopyalanır. Daha sonra Delphi 7.0 programı çalıştırılır. *File* menüsünden *Open* seçeneğine tıklenır. Çıkan penceredeki konum kısmı hedefi olarak, collcontrol klasörünün içerisindeki source klasörü seçilir. Hedefin içerisinde bulunan dosyalardan *CoolCtlsD7.dpk* dosyası seçilerek aç butonuna tıklanır. *Aç* butonu tıklandıktan sonra Resim 3.1'de görünen *Package – CoolCtlsD7.dpk* penceresi açılır. Bu pencerede önce *Compile* butonuna, ardından da *install* butonuna tıklanır. Bu işlemlerin ardından CoolControl paketine ait tüm nesneler yüklenmiş olur.



Resim 3.1. Package – CoolCtlsD7.dpk penceresi

Coolcontrol paketi Delphi 7.0 a yüklendikten sonra, *Tools* menüsünden *Environment Options* seçilerek Resim 3.2'de görünen *Environment Options* penceresi ekrana gelir. Bu penceredeki *Library* sekmesine tıklanır. *Directories* başlığının altında bulunan *Library path* yazı kutucuğunun en sonuna gidilererk ; işareti konulur. Daha sonra ; işaretinin yanına bilgisayara yüklenen Coolcontrol klasörünün içerisindeki Source klasörünün yer adresi yazılır. Örneğin ";D:\CoolControl\Sources".

EK-3. (Devam) Coolcontrol paketinin Delphi 7.0 Programlama diline yüklenmesi



Resim 3.2. Environment Options penceresi

CoolControl işlemi ile ilgili yukarıda açıklanan işlemler yapıldıktan sonra BB3_Simulator.dpr dosyası Delphi 7.0 programında açılırsa BB3 simülatörünün tüm form pencereleri görülebilir. Kaynak kodlar üzerinde değişiklik yapıldıktan sonra program tekrar derlenebilir.

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, adı	: ÇETİN, Gökmen
Uyruğu	: T.C.
Doğum tarihi ve yeri	: 18.05.1979 Elazığ
Medeni hali	: Bekar
Telefon	: 0 (312) 495 39 43
e-mail	: gokmen.cetin@tib.gov.tr

Eğitim

Derece	Eğitim Birimi	Mezuniyet tarihi
Lisans	Gazi Üniversitesi/ TEF Bilg. Sis. Öğrt.	2001
Lise	Bahçelievler Deneme Lisesi	1996

İş Deneyimi

Yıl	Yer	Görev
2001-2004	MEB Nallıhan Ş.Ö.B. ÇPL	Bilgisayar Öğretmeni
2004-2004	MEB GOP And. Mes. Lisesi	Bilgisayar Öğretmeni
2004-2009	MEB Türk Telekom ATL	BT Dal Şefi
2009-….	Telekominikasyon İletişim Başk.	Uzman

Yabancı Dil

İngilizce

Hobiler

Sinema, Müzik, Seyahat, Basketbol, Futbol