Electronic Letters on Science & Engineering 5(1) (2009)
Available online at www.e-lse.org

AN ASSEMBLER and COMPILER PROGRAM DESIGN FOR AN EDUCATIONAL PURPOSE COMPUTER ARCHITECTURE SIMULATOR BEING USED COMPUTER AND ELECTRICAL ENGINEERING EDUCATION

Halit Öztekin[1], Feyzullah Temurtaş[2], Ali Gülbağ[1]
[1]*Sakarya University, Computer Engineering Department, Esentepe Campus, 54187, Sakarya, TURKEY*
[2]*Bozok University, Department of Electrical and Electronics Engineering, 66100 Yozgat, TURKEY*
oztekinhalit@gmail.com

Abstract: In this study, an assembler and Compiler program for an educational purpose computer architecture simulator which is used in Computer Engineering in Sakarya University and Electrical and Electronic Engineering in Bozok University is designed. The students in departments obtained the bit file that the Computer Architecture Simulator requires by writing their programs to realize through this program. The errors in the assembly code writing are eliminated, the compiling time that take very big time is very big time is skeletonized and the example count which is applied during the course is occurred the increment. Also, it is shown how the bit file obtained by writing with the example programs.

Keywords: Microprocessor, Educational Purpose, Assembler, Computer Architecture

BİLGİSAYAR VE ELEKTRİK MÜHENDİSLİĞİ EĞİTİMİNDE KULLANILAN EĞİTİMSEL AMAÇLI BİLGİSAYAR MİMARİSİ SİMÜLATORÜ İÇİN ASSEMBLER ve DERLEYİCİ PROGRAM TASARIMI

Özet: Bu çalışmada, Sakarya Üniversitesi Bilgisayar Mühendisliği ve Bozok Üniversitesi Elektrik-Elektronik Mühendisliği bölümlerinde kullanılan eğitimsel amaçlı bir bilgisayar mimarisi simülatörü için bir assembler ve derleyici programı tasarlanmıştır. Bu program sayesinde bölümdeki öğrenciler, gerçekleştirmek istedikleri programları yazarak Bilgisayar Mimarisi Simülatörünün ihtiyaç duyduğu bit dosyasını kolaylıkla elde etme şansını bulmuşlardır. Manüel olarak yazılan assembly kod yazımındaki hatalar giderilmiş, derleme zamanındaki büyük zaman kayıpları minimuma çekilerek, ders esnasında uygulanabilen örneklerin sayısında artma meydana gelmiştir. Ayrıca bu çalışmada, programın nasıl çalıştığının ve bit dosyasının nasıl elde edildiğine dair örnek programlar yazılarak gösterilmiştir.

Anahtar Kelimeler: Mikroişlemci, Eğitimsel Amaç, Assembler, Bilgisayar Mimarisi

Reference to this paper should be made as follows (bu makaleye aşağıdaki şekilde atıfta bulunulmalı):
H.Oztekin, F. Temurtas, A. Gulbag, 'An Assembler and Compiler Program Design For An Educational Purpose Computer Architecture Simulator Being Used Computer And Electrical Engineering Education', Elec Lett Sci Eng ,vol. 5(1) , (2009), 9-16

1 Giriş

Bilgisayar Mühendisliği ve buna benzer bilim dallarında Bilgisayar Mimarisi ve Organizasyonu dersi bu mühendislik dallarının temelini oluşturmaktadır. Bu bilim dallarından mezun olan bir öğrencinin bir bilgisayar mimarisinin nasıl oluşturulduğu hakkında bilgi sahibi olması gerekmektedir. Ancak bu derslerde gösterilen kavramların teorik olmaktan öteye geçememekte olup, bundan dolayı öğrencilerin zihninde bu kavramlar tam olarak oturmamaktadır. Bu eksikliği gidermek adına eğitimsel amaçlı bir bilgisayar mimarisi

Corresponding author; Tel.: +(90) 354 242 10 01 , E-mail:oztekinhalit@gmail.com

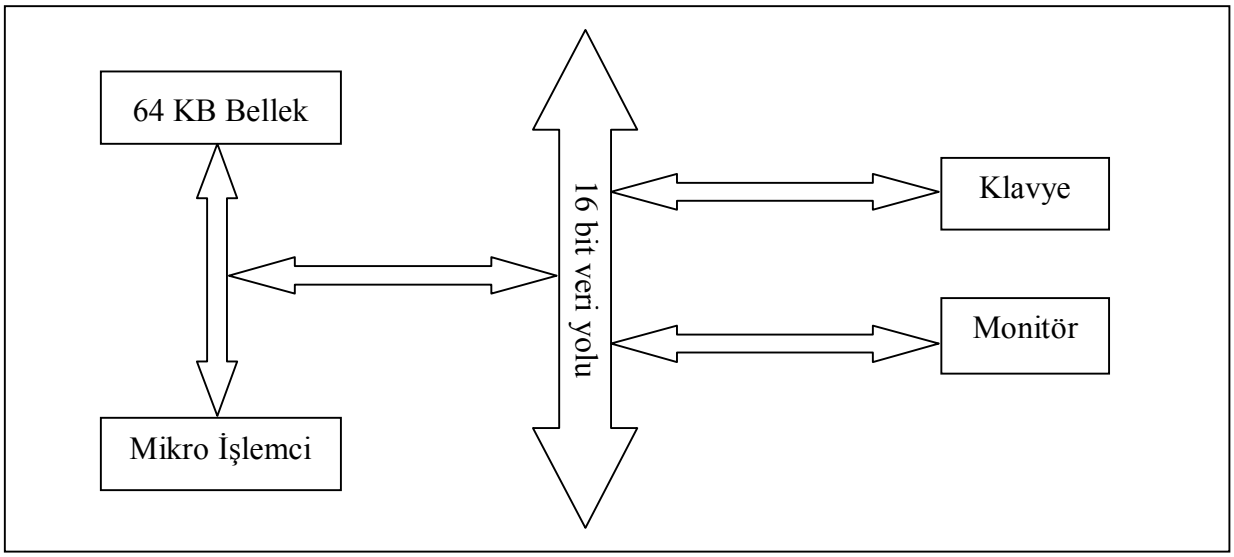
ISSN 1305-8614

simülatörü geliştirilmiştir[1]. Bu simülatör vasıtasıyla öğrenciler bilgisayar mimarisi hakkında detaylı bilgiye sahip olmakla beraber, simülatör bütünüyle donanımsal olarak oluşturulduğundan simülatörün iç yapısı üzerinde değişiklik yaparak veya iç yapısını inceleyerek kendi bilgisayar mimarilerini oluşturabilme yönünde öğrencilere bir fırsat sunulmuştur. Ancak bu bilgisayar mimarisi üzerinde programların çalışabilmesi için mikroişlemcinin belleğine programların yazılıp derlenmesi işlemi manüel olarak yapılması gerekmekteydi. İşte bu sebepten dolayı öğrencilerin programlarını yazıp derleyebilecekleri bir assembler programı oluşturuldu. Bu program sayesinde öğrenciler manüel olarak yazma durumunda karşılaşılan; komutun yanlış opcode'un girilmesi, yanlış adreslere dallanılması, programın belleğin ilgili kısma yazılmaması gibi problemler ortadan kaldırılmıştır.

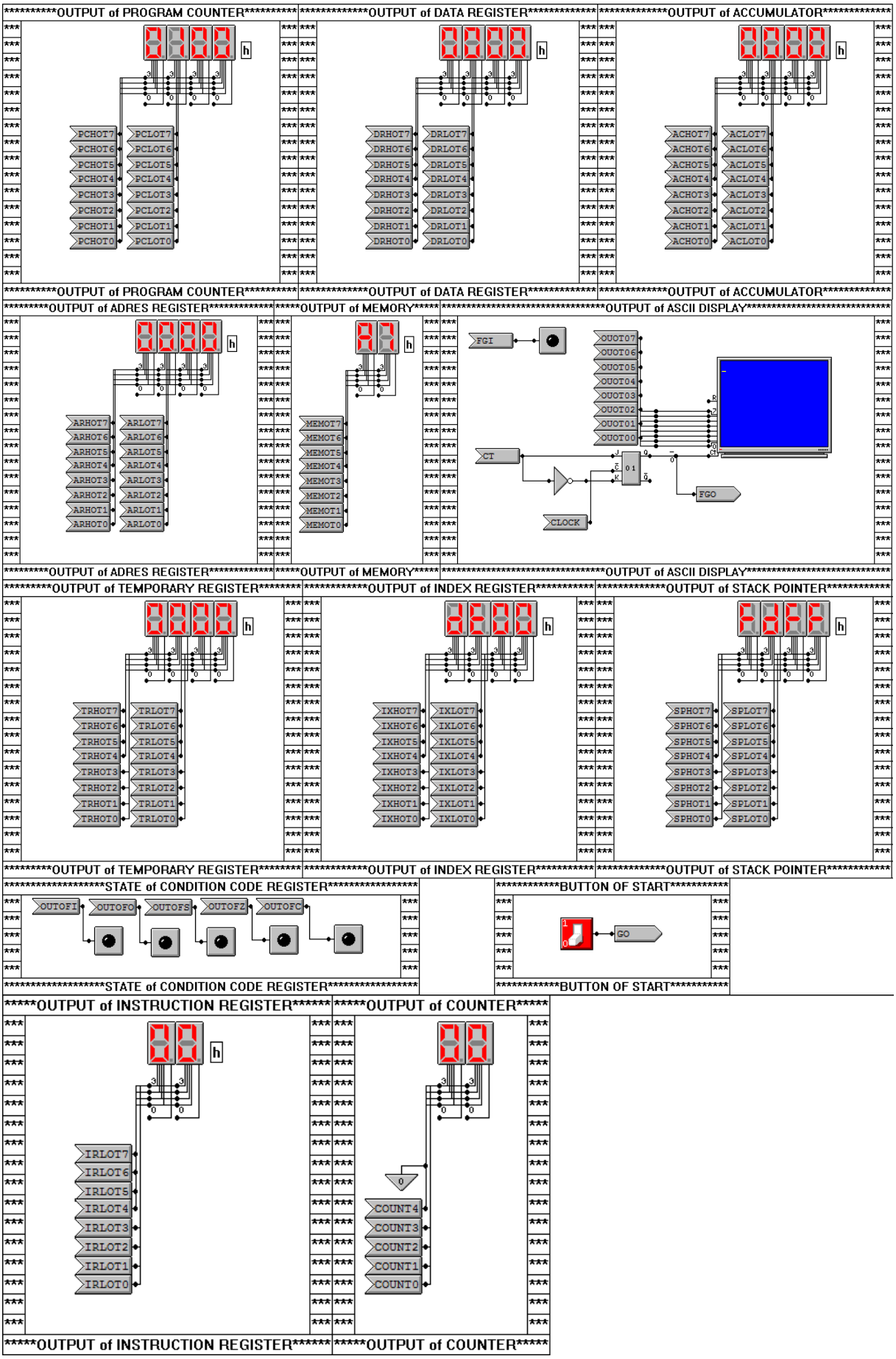
Bu makalenin bundan sonraki bölümleri şu şekilde organize edilmiştir. Bilgisayar mimarisi simülatörü hakkında bilgi Bölüm 2'de, tasarlanan assembler programının tanıtımı Bölüm 3'de, örnek uygulamalar Bölüm 4'de ve sonuç kısmı da Bölüm 5'de yer almaktadır.

2 Bilgisayar Mimarisi Simülatörü

Bilgisayar Mimarisi Simülatörü, bellek ve akümülatör üzerinde işlem yapan 21 adet, indeks ve yığın üzerinde işlem yapan 8 adet, dallanma işlemleri yapabilen 22 adet, durum kod kaydedicisi üzerinde işlem yapabilen 6 adet ve 2 adet giriş-çıkış komutları olmak üzere toplam 59 adet komut icra edebilen 8 bitlik bir bilgisayardır. Bu komutları yerine getirirken kullanmış olduğu altı farklı adresleme modu vardır. Bunlar; derhal(immediate), direkt(direct), dolaylı(indirect), indis(index), göreceli(relative) ve doğal(inherent) adresleme modlarıdır. Bu bilgisayar bu komutları icra ederken 11 adet kaydedici(register) kullanmaktadır. Kullanılan kaydedicilerin bir kısmı 16 bit ve bir kısmı ise 8 bit olarak tasarlanmıştır. Bu elemanların birbirleriyle haberleşmesini sağlayacak veri yolu 16 bitten oluşmaktadır. Adres kaydedicisi 16 bit olduğundan tasarlanan bilgisayarın kullanabileceği bellek alanı 64KB'dır. Tasarlanan bu bilgisayarda göreceli adresleme ve indis adresleme modu kullanan bir komut ile karşılaşıldığında etkin adresi hesaplamak için Aritmetik ve lojik biriminin(ALU) yerine özel bir devre tasarlanmış olup, bu moddaki komutları yerine getirilme zamanı ALU'yu kullanarak etkin adresi hesaplama yöntemine göre altı çevrim(cycle) daha az zaman harcanması sağlanmaktadır. Bu tasarım ilerleyen bölümlerde ayrıntılarıyla ele alınacaktır. Bilgisayar mimarisi simülatörünün blok diyagramı Şekil 1'de, simülatör çalıştığı anda bütün kaydedicilerin içeriğinin gözlenebildiği simülatör başlangıç ara yüzü Şekil 2'de görülmektedir.



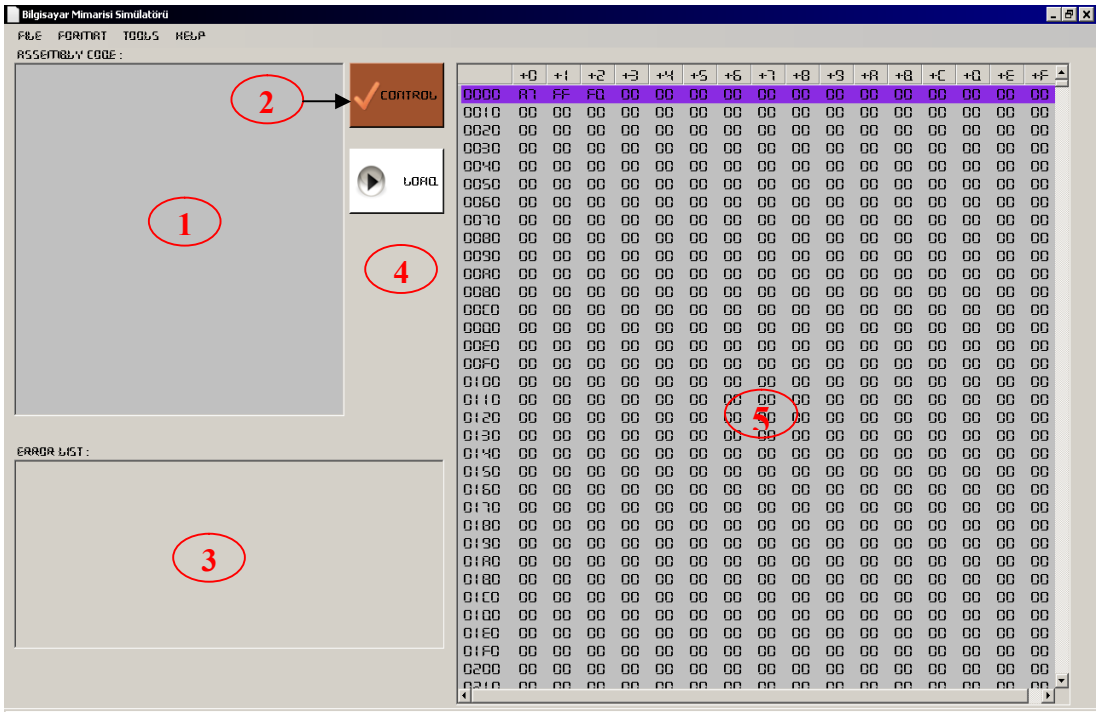
Şekil 1. Bilgisayar Mimarisi Simülatörü Blok Diyagramı



Şekil 2. Simülatörün ilk çalıştığı andaki ekran görüntüsü

3 Tasarlanan Assembler ve Derleyici Programı

Günümüz yüksek seviyeli dillerde yazılan bir programın üzerinde çalıştırılacak işlemcinin sahip olduğu komutlara göre düzenlenmesi gerekmektedir. Bu düzenleme işlemini assembler programları yapmaktadır. Bu işlemin ardından ise derleme işlemi gelmektedir. Yazılan programın derlenmesi işi ise derleyici program tarafından yerine getirilmektedir. Program derlenirken referans olarak, mikro işlemci opcode tablosu kullanılır. Bu derleme işlemini manüel olarak yani opcode tablosuna bakarak yapmak oldukça zahmetli bir iştir. Bu sebepten ötürü, yazılan programların derlenmesi için derleyici programlar kullanılmaktadır. İşte bu çalışmada eğitim amaçlı olarak çalışan Bilgisayar Mimarisi Simülatöründeki mikro işlemci için bir assembler programı Microsoft Visual Studio.NET ortamında tasarlanmış olup, program içerisinde bulunan gömülü derleyici programı vasıtasıyla yazılan program derlenip belleğe yerleştirilmektedir[2]. Assembler programının ara yüzü Şekil 3'de görülmektedir.



Şekil 3. Simülatör için tasarlanan Assembler ve Derleyici programı arayüzü

1: Yazılmak istenen programın assembly kodlarının yazıldığı alan.
2: Yazılan assembly kodlarda bir hata bulunup bulunmadığını denetleyen "Control" butonu.
3: Assembly kodunun varsa hatalarını görüntüleyen bilgi ekranı.
4: Assembly kodun opcode'lara dönüştürülerek "E" bellek alanına yerleştirilmesi.
5: İşlemcinin kullanmış olduğu bellleğin iç yapısı. Belleğin ilk üç gözü kesme için kullanılmıştır. Kesme geldiğinde program FFF0 adresine dallanır.

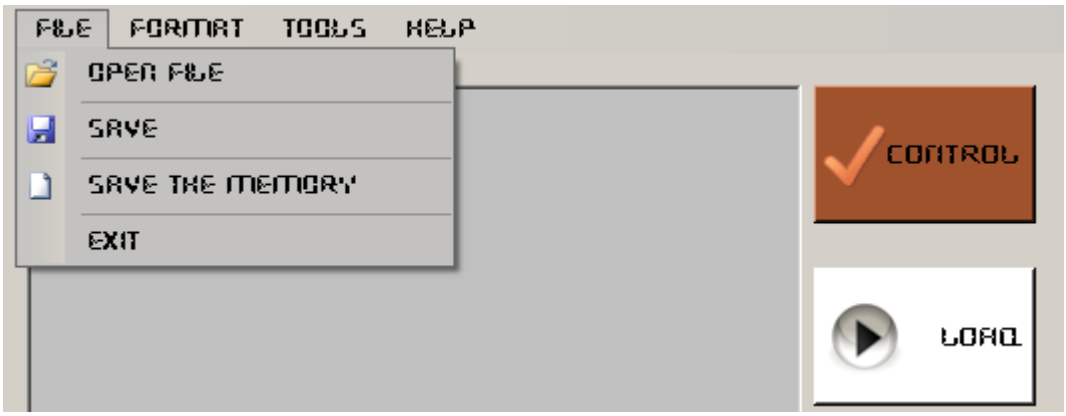
3.1 Program Menüleri

Yazılan Assembly ve Derleyici programı dört adet menüden oluşmaktadır. Bunlar; File, Format, Tools ve Help menüleridir.

3.1.1 FILE Menüsü

File menüsü dört ana kısımdan oluşmaktadır;

OPEN FILE **:** Metin editöründe yazılmış bir assemly kodu programın "A" alanına aktarma işlemini yerine getirir.
SAVE **:** "A" alanında yazılmış programı bilgisayara text olarak saklamak için kullanılır.
SAVE THE MEMORY: "E" alanında belleğin içeriği bir text dosya olarak bilgisayara kaydetmek için kullanılır. Bu dosya aynı zamanada Bilgisayar Mimarisi Simülatöründeki işlemcinin belleğini oluşturur.
EXIT **:** Programdan çıkışı sağlar.

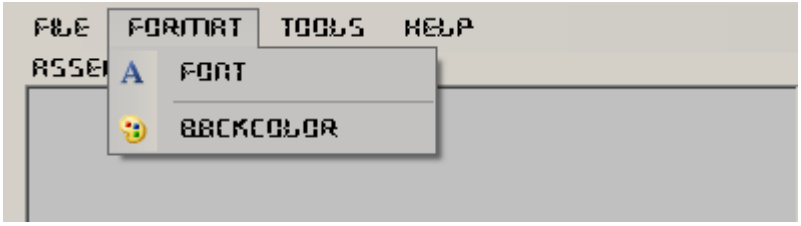


Şekil 4. Programın "File" menüsü.

3.1.2 Format Menüsü

Format Menüsü iki ana kısımdan oluşmaktadır.

FONT **:** Yazılan Assembly kodunun yazı tipini değiştirir.
BACKCOLOR **:** Programdaki "E" alanının arkaplanının rengini ayarlar.

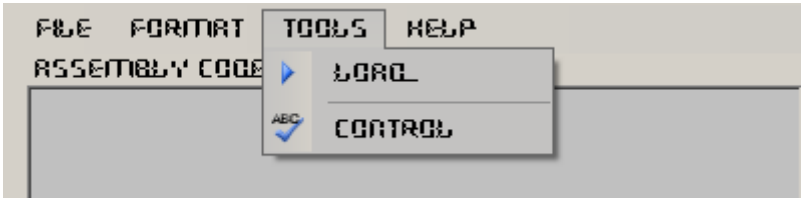


Şekil 5. "Format" Menüsü

3.1.3 Tools Menüsü

Tools menüsü iki kısımdan oluşmaktadır.

LOAD **:** Yazılan programı derleyerek "E" alanındaki belleğe yerleştirir.
CONTROL **:** Assembly kodda bir hata bulunup bulunmadığını denetler.



Şekil 6. "Tools" Menüsü

3.1.4 Help Menüsü

Help menüsü 3 bölüm içermektedir.

CONTACT US ***:*** Programın iletişim bilgileri yer almaktadır.
HELP ***:***Assembly kod yazımında kullanılacak komut listesi ve adresleme modlarını temsil eden simgelerin yer aldığı bilgileri içermektedir.
ABOUT ***:*** Programın versiyon bilgisi yer almaktadır.

Şekil 7. "Help" Menüsü

3.2 Program Kullanımı

Programın Assembly kod alanında program yazılırken uygulancak kurallar şunlardır:

1. Bilgisayar Mimarisi Simülatörü yukarı değinildiği üzere altı farklı adresleme modunu kullanmaktadır. Assembly Code yazımının yapıldığı "A" alanında bu adresleme modlarını temsil eden simgelerin kullanılması gerekmektedir. Bu simgelerden; "#" ivedi adresleme modunu, "$" direkt adresleme modunu, "@" dolaylı adresleme modunu, "%" indis adresleme modunu, "*" göreceli adresleme modunu temsil ederken doğal adresleme modunu kullanırken herhangi bir simge kullanmaya gerek yoktur.

2. Simülatör, akümülatör ve bellek komutları, yığın ve indis komutları, sıçrama ve dallanma komutları ve giriş-çıkış komutları olamak üzere 4 ana bölümden oluşan komut setine sahiptir[3]. Bu komut setinden yararlanırken ivedi, direkt ve dolaylı adresleme modunu kullanan bir komut kullanılmak istendiğinde, "A" alanına girilecek komut şu formatta olmalıdır.

"komut"*boşluk****"adresleme modunu temsil eden simge"dört basamaklı hexadecimal sayı"H"***

Örnek:

İvedi adresleme modunu kullanan ADD komutu : ADD #1234H
Direkt adresleme modunu kullanan ADD komutu : ADD $1234H
Dolaylı adresleme modunu kullanan ADD komutu : ADD @1234H

Eğer kullanılan komut indis ve göreceli adresleme modunu kullanıyorsa;

"komut"*boşluk****"adresleme modunu temsil eden simge"iki basamaklı hexadecimal sayı"H"***

Örnek:

İndis adresleme modunu kullanan ADD komutu : ADD %12H
Göreceli adresleme modunu kullanan ADD komutu : BRA *12H

Kullanılan komut doğal adresleme modunu kullanıyorsa;

"komut"

yapısında olmalıdır.

Örnek:

INCR

4 Örnek Uygulama

Yüksek seviyeli dillerde çok kullanılan aşağıdaki program parçacığını simülatörde gerçeklemeye çalışalım.

```
a=5;
for i=1 to 10
{
a=a+i
}
```

Bilgisayar mimarisi simülatöründe değişkenler veri segmentinde saklanmaktadır. Bu segmentin başlangıç adresi DE00h'dır. Aşağıdaki assembly kodda koyu siyahla yazılmış ve /* ile başlayıp */ ile biten açıklamalar yazılan programın anlaşılabilirliğini artırmak için konulmuştur. Bunlar çıkarıldıktan sonra kalan kısım Assembly ve Derleyici programının "A" alanına kopyalanacaktır.

0100h:LDA #0005h /* Akümülatöre "a" değişkenine atılacak olan decimal 5 değeri atılıyor.*/
0103h:STA $DE00h /* Akümülatördeki 5 değeri "a" değişkeni için veri segmentinin ilk iki gözü ayrılarak, burada saklanıyor*/
0106h:LDA #0001h /* Akümülatöre "i" değişkenine atılacak olan decimal 1 değeri atılıyor.*/

0109h:STA $DE02h /* Akümülatördeki 1 değeri "i" değişkeni için veri segmentinin 3 ve 4. gözü ayrılarak, burada saklanıyor*/
010Ch:LDA #000Ah /* Akümülatöre "i" değişkeninin alabileceği maksimum değer olan decimal 10 değeri atılıyor.*/
010Fh:STA $DE04h /* Akümülatördeki 10 değeri veri segmentinin 5 ve 6. gözü ayrılarak, burada saklanıyor*/
Toplam: 0112h:LDA $DE00h /* "a" değişkeni veri segmentinden alınıp akümülatöre atılıyor.*/
0115h:ADD $DE02h /* "a" değişkeni ile i değişkeni toplanıyor.*/
0118h:STA $DE00h /* toplam değer "a" değişkeninin saklı tutulduğu yere kaydediliyor*/
011Bh:LDA $DE02h /* "i" değişkeni veri segmentinden alınıyor.*/

011Eh:SUB $DE04h /* decimal 10'dan i değişkeni çıkartılıyor*/
0121h:BZR *09h(son) /* eğer i değişkeni decimal 10 değerine ulaşmışsa "son" etiketine dallanıyor*/
0123h:LDA $DE02h /* "i" değişkeni veri segmentinden alınıyor*/
0126h:INCR /* "i" değişkeni bir artırılıyor*/
0127h:STA $DE02h /* "i" değişkeni saklanıyor */
012Ah:BRA *E6h(toplam) /* "toplam" etiketine dallanılıyor/*
Son: 012Ch:HLT

5 Sonuç

Bu çalışma ile Sakarya Üniversitesi Bilgisayar Mühendisliği ve Bozok Üniversitesi Elektrik-Elektronik Mühendisliği bölümünde eğitim materyali olarak kullanılan Bilgisayar Mimarisi Simülatörü için Assembly ve Derleyici program geliştirilmiştir. Bu program sayesinde öğrenciler işlemci üzerinde koşturacakları programları hatasız bir şekilde yazma ve derleme işlemi için harcanan zaman minimuma indirilmiştir. Bu materyali kullanan öğrenciler arasında yapılan anket çalışmasında öğrencilerin büyük bir kısmı: "Bu program sayesinde bizim üzerinde koşturacağımız programların çeşitliliğini artırmada faydalı olmuştur" ifadesini kullanmışlardır. Sakarya Üniversitesi Bilgisayar Mühendisliği bölümünde yapılan 116 öğrenci üzerinde yapılan araştırma ortaya çıkan aşağıdaki tablo bu durumu özetlemektedir.

Tablo1. Programın sağlamış olduğu özellikler

	Programdan* Önce	**Programdan* Sonra**
Hatasız Kod Yazma Yüzdesi	%5	%100
Derleme Zamanı**	4–7 dk.	5 sn
Program Çeşitliliği	2–3/ders	9–10/ders

* : Bilgisayar Mimarisi Simülatörü için Tasarlanan Assembly ve Derleyici Programı
** : Uygulama–1 kısmında verilen programın derlenme zamanları.

Bu tablodan da görüleceği üzere bir programı hatasız olarak yazan öğrenci yüzdesinin %5'lerden %100'lere çıktığı görülmüştür. Çünkü program, hatasız bir şekilde kod yazılmadan derleme işlemini yapmamaktadır. Derleme zamanı uygulama-1'de gösterilen örnek için dakikalar mertebesinden saniyeler mertebesine inmiştir. Yine bir deste gösterilen örnek uygulamaların sayısının yaklaşık olarak %400 kadar artması tablodan çıkarılabilecek diğer bir sonuçtur. Çünkü derste bir uygulama kod parçacığının kalem kâğıtla yazılıp, mikroişlemci kod tablosuna bakarak opcode'larını yazma işlemi oldukça zaman aldığından ders esnasında fazla uygulama şansı olmamaktadır.

Dallanma komutları kullanıldığında, programın dallanacağı adresi manüel olarak hesaplama gereği bu programın eksik yanını oluşturduğundan bir sonraki çalışmada programın dallanacağı yerlere etiketler konularak bu eksiklik bertaraf edilecek olup, programın kullanımını artıracak ilaveler yapılacaktır.

References

1. H. Öztekin, Bilgisayar Mimarisi Simülatörü Tasarımı, Sakarya Üniversitesi Mühendislik Fakültesi Bilgisayar ve Bilişim Mühendisliği, M.S. Thesis, SAU, 2009
2. Bilgisayar Mimarisi Simülatörü için geliştirilen Assembler ve Derleyici Programı, http://eem.bozok.edu.tr/database/1/Assembly_Program_For_BZK_SAU.rar
3. Bilgisayar Mimarisi Simülatörünün kullandığı Komut Seti, http://eem.bozok.edu.tr/database/1/Command_Set_for_Computer _Architecture_Simulator.rar