T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

8051 MİKRODENETLEYİCİSİ İÇİN İŞLEVSEL TABANLI BENZETİM PROGRAMI TASARIMI

YÜKSEK LİSANS TEZİ

Necat GÜNEY

Enstitü Anabilim Dalı : ELEKTRONİK BİLGİSAYAR EĞİTİMİ

Bu tez 03/08/2006 tarihinde aşağıdaki jüri tarafından Oybirliği ile kabul edilmiştir.

Yrd. Doç. Dr.	Prof. Dr.	Doç. Dr.
A. Turan ÖZCERİT	Abdullah FERİKOĞLU	Nejat YUMUŞAK
Jüri Başkanı	Üye	Üye

TEŞEKKÜR

Tez çalışmalarıma başladığım günden bugüne kadar gösterdiği destek ve anlayıştan dolayı tez danışmanım Sn. Yard. Doç. Dr. Ahmet Turan ÖZCERİT'e, tez çalışmamın oluşumunda yardım ve desteklerini esirgemeyen Bolu Merkez İzzet Baysal Anadolu Teknik Lisesi okul idaresine ve Bilgisayar Bölümü'nde beraber çalıştığımız meslektaşlarıma teşekkür ederim.

İÇİNDEKİLER

SİMGELER VE KISALTMALAR LİSTESİ

#	: Sağındaki değerin veri olduğunu gösterir
$	: Sağındaki değerin ondalık sayı veya adres olduğunu gösterir
@	: Sağındaki verinin sekiz tabanlı sayı olduğunu gösterir
O	: Önündeki verinin sekiz tabanlı bir sayı olduğunu gösterir
B	: Önündeki değerin ikilik sayı olduğunu ifade eder
H	: Önündeki değerin onaltılık tabanda bir değer olduğunu gösterir
%	: Sağındaki değerin ikilik sayı olduğunu ifade eder
D	: Önündeki verinin ondalık düzende olduğunu gösterir

ACC,A	: A kaydedicisi, akümülatör
B	: B kaydedicisi
SFR	: Özel Fonksiyon Kaydedicileri
PSW	: Program Durum Kaydedicisi
DPTR	: Data Pointer-Veri İşaretçisi
SP	: Stack Pointer-Yığın İşaretçisi
PCON	: Power Control-Güç Kontrol
IE	: Interrupt Enable-Kesme Yetkilendirme
IP	: Interrupt Priorty-Kesme Öncelik
TCON	: Timer Control-Zamanlayıcı Kontrol
TMOD	: Timer Mod-Zamanlayıcı Mod
SCON	: Serial Control-Seri Kontrol
SBUF	: Serial Buffer-Seri Tampon
Rn	R0-R7'ye kadar 8 adet kaydedici

ŞEKİLLER LİSTESİ

TABLOLAR LİSTESİ

ÖZET

Anahtar Sözcükler: Mikrodenetleyici simülasyonu, 8051 simülatörü, fonksiyonel simülatör, 8051

Bu çalışmada, endüstride neredeyse her alanda kullanılan 8051 mikrodenetleyicisinin, öğretiminin kolaylaştırılmasını sağlayan simülatör tasarımı amaçlanmıştır. Öğretim ve tasarım amaçlı hazırlanan bu simülatörün yapımında diğer simülatörlerden farklı olarak, assembly kodların adım adım çalıştırılması sırasında ilgili komutun etkilediği yazmaçların ve bayrakların takibini kolaylaştıran bir yapı kullanılmıştır. İkinci farklılık ise, mikrodenetleyici portlarına kolaylıkla bağlanabilen (anahtar, buton, yedi parçalı gösterge, led) elemanları içeren bir araç kutusu eklenmiştir. Bu araç kutusundaki elemanların portlara bağlantısı assembly kodları derlendikten sonra da yapılabilmektedir. Böylece tasarlanan simülatör bir mikrodenetleyici deneme kartı gibi de kullanılabilmektedir. Hazırlanan simülatörün bu iki özelliği ile 8051 mikrodenetleyicisinin yapısının daha kısa sürede ve kolay anlaşılması sağlanmaya çalışılmıştır.
.

THE DESIGN OF A FUNCTIONAL BASED SIMULATOR FOR THE 8051 MICROCONTROLLER

SUMMARY

Key words: Microcontroller simulation, 8051 simulator, functional simulator, 8051

In this thesis, a functional based simulator has been designed to facilitate the education and engineering of 8051 microcontroller which is used widely in today's industry. In the realization of the simulator, a PC-based program is developed in a way to make easy to follow the contents of the registers and flags of the running an 8051-coded assembly program. Another important contribution fulfilled is a toolbox which can easily be connected to the ports of the microcontroller using as switch, button, display, etc. The elements in this toolbox can also be connected after compilation period. With the help of this facility, the simulator can be used as a virtual microcontroller development board. These properties of the simulator make it easy to understand the internal architecture of the 8051 microcontrollers.

BÖLÜM 1. GİRİŞ

Mikrodenetleyici, bir bilgisayar sisteminin temel özelliklerini içeren tek bir silikon kılıf içerisinde toplanmış bir tümdevredir. Çoğu zaman günlük yaşamda kullandığımız birçok cihaz içerisinde gizlenen mikrodenetleyicilerden birçoğumuzun haberi dahi olmaz [1]. Mikrodenetleyiciler, ucuz maliyetleri ve programlanabilme özelliğinden dolayı endüstrinin her alanında kullanılmaktadır. Günlük hayatta kullandığımız her cihazda neredeyse bir mikrodenetleyici vardır. Evimizdeki buzdolabı, çamaşır makinesi, bulaşık makinesi, telefonlardan tutun da bindiğimiz otomobillere kadar geniş bir kullanım alanı olan bu tümdevre elamanı ön lisans, lisans ve yüksek lisans eğitimi sırasında ders olarak okutulmaktadır.

Özellikle günümüzde elektrik-elektronik mühendisliği eğitimi alanında oldukça yaygın olarak kullanılan tasarım araçlarından birisi de mikrodenetleyicilerdir. Geçmişe nazaran oldukça fazla sayıda olan üretici firma ve bunların sonucu ortaya çıkan onlarca sistem geliştirme yazılımları bu alanda çalışan tasarımcı sayısını hızla artırmıştır. Daha da önemlisi, yeni teknolojilerle zenginleşen ve maliyetleri son on yıl içinde hızla düşen mikrodenetleyiciler, tasarımcılar için büyük bir cazibe odağı haline gelmiştir [1].

M. R. Smith, ve M. Cheng [2], "Mikroişlemci laboratuarlarında ve çalışma kağıtlarında sanal (simule edilmiş) donanım aygıtları kullanımı", adlı çalışmasında, elektromekanik deney setlerinin; birçok elektronik malzeme kullanılarak yapıldıkları için yüksek maliyete sahip eğitim araçları olduğunu ve bu tür laboratuarların kurulumunun zorluğunu anlatıyor. Yüksek maliyetten dolayı laboratuar sayısı az olduğundan, öğrenciler ders dışındaki boş zamanlarında istedikleri zaman laboratuarları kullanamamaktadırlar. Çünkü laboratuarlar belli bir programa göre öğrenci gruplarının kullanımına sunulmaktadır. Bu nedenle de öğrenciler yazdıkları programları deneme şansını bulamadıklarından bahsediyor.

Ülkemizdeki kalabalık sınıflar düşünülürse bu maliyetin daha da yükseleceği kaçınılmaz bir sonuç olarak karşımıza çıkar. Bu tür laboratuarlarda en az beş altı öğrenci bir deney seti ile çalıştığından eğitim kalitesini olumsuz yönde etkilemesi söz konusudur. Maliyeti düşürmek için okullarda bilgisayar öğretimi için hali hazırda kullanılan laboratuarlarda, bilgisayarlara öğretimi yapılacak mikroişlemci veya mikrodenetleyici dersi için kurulacak bir simülatör programı uygun bir çözüm olacaktır. Günümüzde okullardaki genel amaçlı bilgisayar laboratuarları genelde öğrencilerin ihtiyacını karşılayacak sayıda olduğundan dolayı öğrenciler ders dışında da bu laboratuarları kullanabilmektedirler. Hatta günümüzde birçok öğrenci kendisine ait bir dizüstü veya masaüstü bilgisayara sahiptir. Bu durum, öğrencilere neredeyse istedikleri her zaman simülatör programını kullanma imkanı sunacaktır. Tasarlanan 8051 mikrodenetleyici simülatörü bu amaçla kullanılabilir. Hazırlanan simülatör ücretsiz dağıtacağından dolayı maliyet olarak sadece hazır bilgisayar laboratuarlarının veya bilgisayarın bakım masrafı karşımıza çıkacaktır.

C. W. Caldwell, D. L. Andrews, ve S. S. Scott [3]'un yaptığı "Sınıf kullanımı için grafiksel mikrobilgisayar simülatörü" adlı çalışmada, elektromekanik deney setleri birçok elektronik ve mekanik parçalar meydana geldiğinden, sık arıza yapabilmekte ve bu arızaları giderebilecek elemanlara ihtiyaç olduğundan, deney setlerinin kullanımının da zor olduğundan bahsedilmiştir. Hazırladıkları simülatörle deney setlerin birçok olumsuz etkinsin ortadan kaldırıldığını ve mikroişlemcinin genel yapısını ve giriş/çıkış yapısının öğretiminin kolaylaştırıldığını söylemişler. Grafiksel simülatörün işlemcinin iç yapısını ve çalışmasını gözlemleme imkanı sağladığını ve öğretimi olumlu yönde etkilediği bildirilmiştir.

W. P. Lovergrove [4] "Mikroişlemci eğitim simülatörü" çalışmasında öğrencilerin laboratuar çalışmalarını okul dışında devam ettirmelerini amaçlayan bir simülatör tasarlamıştır. Program yazımı için bir metin düzenleme editörü kullanılmamakta ve simülatör grafik tabanlı bir işletim sisteminde yapıldığı halde görsellik arzu edilen seviyede değildir. Simülatörün tasarım amaçlarından birisinin de deney setlerinin olmadığı ortamlarda da öğrencilerin bu simülatörü kullanarak kod yazabilme imkanın sağlandığından bahsedilmiştir. Deney setlerinde program kodlarının elle tek tek

girilmesi gerektiğini ve yanlışlık olduğunda baştan tekrar kodların girilmesin zaman kaybına yol açtığını ve bu sorunun simülatör ile aşıldığından bahsedilmiştir.

Victor Giurgiutiu, Jed Lyons, David Rocheleau ve Weiping Liu [5] tarafından yapılan "Güney Carolina Üniversitesi makine mühendisliği öğrencileri için mekatronik ve mikrodenetleyici eğitimi" isimli projede makine mühendisliği öğrencileri mikrodenetleyiciler dersini almadan önce programlama ile ilgili herhangi bir eğitim almadıklarından yazılım konusunun öğrenciler için zorlayıcı olduğu belirtilmektedir. Mikrodenetleyiciler için simülasyon yazılımının kullanımı öğrencilerin bu zorluğu aşmasında kolaylık sağladığı belirtilmektedir. Seçtikleri simülatör programının maliyetinin düşük olduğunu ve bu programla 68HC11 için kod yazılabildiğini /düzenlenebildiğini, kodların derlenip simülasyonunun yapılabildiğinden bahsedilmektedir. Programın hata ayıklama işlemlerini yapma kabiliyeti ile beraber bu simülatörün windows tabanlı çalıştığını ve cpu, ram, rom ve giriş çıkış portlarının durumunu gösterebildiğini ve aynı zamanda simülatör üzerinde timer, adc, paralel port, seri port ve g/ç pinlerinin durumlarını da göstermektedir.

Hata ayıklama yapılırken grafik arayüz sayesinde tüm kaydedicilerin bellek içeriklerinin ve pinlerin durumu gözlenebildiğini, hatta program çalışıyorken bile gözlenebilmektedir. Program çalıştırılırken de harici elemanlar pinlere bağlanabilmektedir.

Bu yaklaşımla öğrencilerin mikrodenetleyiciler hakkındaki bilgi seviyelerini daha da ileriye götürdüklerini ve mikrodenetleyicilerin yapısını incelemekten ziyade çeşitli uygulamalarda mikrodenetleyici kullanımına yoğunlaştıkları bildirilmektedir.

Mehrdad Reshadi, Prabhat Mishra, Nikil Dutt [6], "Komut setli derleyici simülasyon: Esnek komut seti ve hızlı bir teknik için", adlı çalışmasında, Yorumlayıcı Simülatörlerle Hızlı Komut Simülatörü'nün avantajlı yönlerini birleştirerek Komut Soyutlama Simülatörü adında yeni bir teknik geliştirmiştir. Yorumlayıcı Simülatörler, yavaş çalışırlar, daha ayrıntılı ve kullanım esnekliği olan simülatörlerdir. Daha çok öğretim amaçlı kullanılırlar. Hızlı Komut Simülatörleri ise tasarım için kullanılır ve hızlı çalıştıkları için çalışma anında müdahale etmek zordur, esnek bir kullanıma sahip de-

ğillerdir.

Mehrdad Reshadi ve arkadaşları bu iki tekniği birleştirerek hem Hızlı Komut Simülatörlerinden %40 daha hızlı çalışan ve hem de Yorumlayıcı Simülatörler gibi esnek kullanıma sahip Komut Soyutlama Simülatörü adını verdikleri bir teknik kullanmışlardır.

Bülent TÜRKELİ [7], "Intel MC8051 mikrodenetleyicisinin PC'de simülasyon ile eğitimi" isimli yüksek lisans tezinde 8051 için 1995 yılında Turbo Pascal programlama dilini kullanarak bir simülatör hazırlamıştır. Bu simülatör DOS işletim sisteminde çalışmakta o günün şartlarına göre oldukça iyi tasarlanmıştır. TÜRKELİ, mikroişlemci ve mikrodenetleyicilerin endüstride çok sık kullanıldığını ve bu yüzden elektronik-bilgisayar eğitiminde ders olarak okutulduğunu, ancak bu eğitimde kullanılacak deney setlerinin ve donanımların az ve pahalı olduğunu belirtmektedir. Bu tür teknoloji eğitiminde laboratuar yetersizliği ve uygulama zorluğu sorunları ile karşılaşılmaktadır. Bu nedenlerden dolayı mikroişlemcilerin PC tabanlı sistemlerde simülasyon yazılımı kullanılarak eğitim ve laboratuar uygulamalarının geliştirilmesi düşünülmüştür. Böylelikle herhangi bir geliştirme setine ihtiyaç duymadan standart donanımlı bir bilgisayar laboratuarı, simülasyonu yazılan farklı işlemciler için aynı zamanda mikroişlemci laboratuarı olarak da kullanılabileceğini belirtilmiş.

İbrahim Hakkı SÜSLÜ [8], "8096 Mikrodenetleyicisinin PC'de simülasyon ile eğitimi", isimli yüksek lisans tezinde 8096 için 1995 yılında bir simülatör sunmuş. SÜSLÜ, MCS-96 mikro denetleyicisinin simülatörünün daha ziyade eğitim amaçlı olarak geliştirildiğini belirtmektedir. Eğitim amaçlı geliştirildiği için bu çalışmada MCS-96 mikrodenetleyicisinin komutlarının geniş bir yardım bölümü hazırlamıştır. Simülatörün kullanıcı ara yüzü ile kullanıcının gözüne hitap etmesi ve sıkmamasına özen göstermiştir. Kullanımı kolay ve sistem geliştirmeden çok Intel MCS-96 mikrodenetleyicisinin PC tabanlı sistemlerde simülasyonunun gerçekleştirilmesi için tasarlamıştır. Bu simülatör sayesinde MCS-96 mikrodenetleyiciye sahip olmayan, fakat bu işlemciyi hem öğrenmek hem de program yazıp, yazılan programı denemek isteyen kullanıcılar dikkate alınmıştır. MCS-96'nın komut setini ve kaydedici yapısını taklit eden SIM8096 adı verilen yazılım Turbo Pascal kullanılarak geliştirilmiştir.

Mustafa NARTKAYA [9], “Intel 8085 mikroişlemcisinin simülasyonu” adlı yüksek lisans tez çalışmada, bilgisayarın çalışma sistemini kavramanın en kolay yolunun mikroişlemci yapısının iyice anlaşılmasıyla olacağını belirtmiştir. Mikroişlemci eğitiminin okullarda deney setleri kullanılarak verildiğini ve bu yöntemin pahalı bir yöntem olduğunu aktarmaktadır. Deney setlerinin sayının öğrenci sayısına kıyaslandığında yetersiz kaldığını ve bu yüzden de bu eğitimin istenilen düzeyde olmadığını belirtmektedir. Hazırlanacak mikroişlemci simülatörünün buna bir alternatif oluşturacağı vurgulanmıştır.

Serdar KÜÇÜK [10], “PIC16C65 Serisi mikrokontrolörün PC’de simülasyon ile eğitimi”, adlı yüksek lisans çalışmasının 1998 yılında tasarlamış ve simülatörü Turbo Pascal programlama dilinde hazırlamıştır. Simülatörün maliyet açısından çok ucuz olduğunu ve kullanıcılara zaman kazandıracağını ileri sürmüştür.

Nurettin TOPALOĞLU [11], “PC Tabanlı fonksiyonel mikroişlemci simülatörü tasarımı ve gerçekleştirilmesi” isimli doktora çalışmasında 6502 mikroişlemcisi için bir simülatör tasarlamıştır. Bu çalışmada, teorik derslerin laboratuar deney uygulamalarında elektromekanik mikroişlemci setleri kullanıldığını ve uygulamanın da birçok dezavantajını göstermiştir. Bu tür deneylerin genelde pahalı olduğu, deneylerin yalnızca laboratuarda yapılabildiği, programlama ve test aşamalarının genelde zaman alıcı olduğunu ve bu tür sistemlerde hataların tespiti ve giderilmesi zor olduğu konusu üzerinde durmaktadır. Mikroişlemci simülatörlerinin genelde daha ekonomik olduğunu, programlama hataları kolayca geliştirme aşamasında sezilebildiğini, bunlara ek olarak da, standart bir PC üzerinde yürütülen grafiksel mikroişlemci animasyonu sayesinde program yürütümünün etkilerinin gözlemlenebildiği konusunu vurgulamıştır.

Tasarımı yapılan mikrodenetleyici simülatörü diğer mikroişlemci ve mikrodenetleyici simülatörlerinden farklı olarak görsellik ve kullanım kolaylığının ön plana çıkarılmaya çalışılmıştır. Bu simülatör 8051’in standart yapısını ve tüm komutlarını simüle edecek şekilde tasarlamıştır.

8051 simülatör sekiz ana bölümden meydana gelmektedir. Simülatör üzerindeki iş-

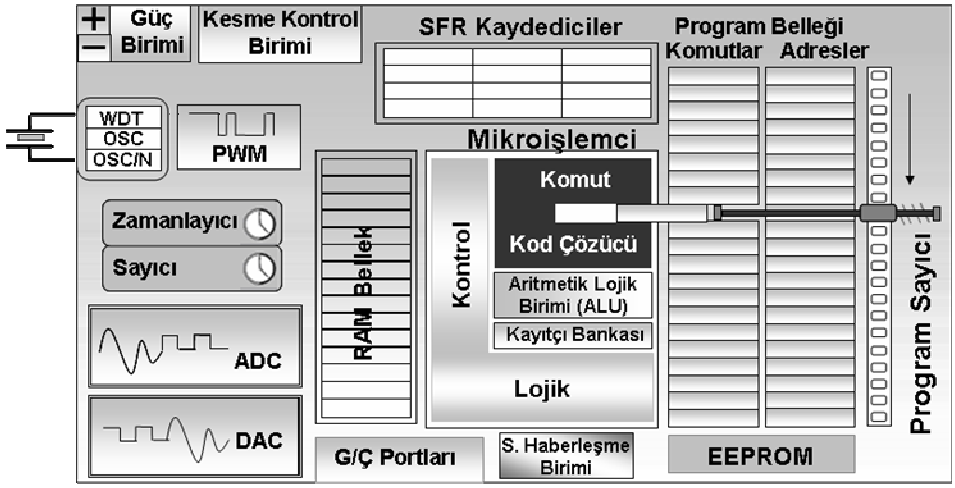
lemler menü seçenekleri kullanılarak veya bu seçeneklerin kısa yol tuş bileşimleri kullanılarak yapılmaktadır. Bu bölümler sırayla:

- Kod yazım ve düzenleme penceresi
- Kod derleme ve hata takip penceresi,
- Program çalıştırma ve takip penceresi,
- Dahili RAM bellek,
- Harici RAM bellek,
- Özel fonksiyon saklayıcıları,
- Program belleği,
- Giriş/Çıkış Aygıtları,

Bu tez çalışması beş ana bölümden meydana gelmektedir. Birinci bölümde simülatörle ilgi genel bilgiler verilmiştir. Diğer simülatörlerle ilgili bazı değerlendirmeler yapılmıştır. İkinci bölümde, mikrodenetleyicilerin yapısı ve çalışma şekli anlatılmıştır. Üçüncü bölümde, simülasyonla ilgili tanımlar verilmiştir ve simülasyonun gerekliliğine vurgu yapılmıştır. Dördüncü bölümde ise simülatörün yapısı, genel kullanım şekli ve çalışma prensibi anlatılmıştır.

BÖLÜM 2. HEDEF MİKRODENETLEYİCİ VE MİMARİSİ

Mikrodenetleyici, tek bir silikon kılıf üzerinde toplanmış entegre devreye denir. Bir mikrodenetleyici tümdevresinde bulunan hafıza ve giriş/çıkış alt sistemleri, bu işlemcilerin birçok uygulama içinde, gömülü olarak doğrudan ve tek başına, mikroişlemcilere göre çok daha basit ve ucuz arabirim teknikleriyle, kontrol amaçlı olarak kullanılmalarını sağlar. Günümüzde mikroişlemciler ev mikrodenetleyici teknolojinin vazgeçilmez birimleri olup, endüstride sayısız alanlarda kullanılmaktadır. Bugün mikroişlemci ve mikrodenetleyici üreten pek çok firma bulunmaktadır. Bunların en önemlileri Philips, İntel ve Motorola firmalarıdır.



Şekil 2. 1. Mikrodenetleyicinin blok diyagramı [1]

Mikrodenetleyici, otomobil sanayisinde çok yoğun olarak kullanılan devre elemanlarından biridir. Araç motorunun değişik noktalarına yerleştirilen algılayıcılardan gelen sinyallerin değerlendirilip motorun çalıştırılmasından, elektrik donanımın kontrol edilmesine kadar kontrol gerektiren her aşamada kullanılmaktadır.

Mikrodenetleyiciler motor kontrolü, fotoğraf makineleri, robotlar, kişisel haberleşme cihazları, merkezi klima sistemleri, bilgisayar kontrollü ulaşım araçları, faks, fotokopi makineleri ve yazıcılar, radyo, teyp, tv, vcd'ler, elektronik beyaz eşyalar, lcd mo-

nitörler, fabrika otomasyonu, biyomedikal cihazlar, oyuncaklar, askeri cihazlar, elektronik bilet uygulamaları (AKBİL) gibi birçok uygulamada kullanılmaktadır.

2.1. Mikrodenetleyici ve Mikroişlemci Karşılaştırılması

Mikroişlemci ile mikrodenetleyici arasındaki farklardan bahsedebilmesi için ikisinin de tanımlarını yapmamız gerekmektedir. Mikroişlemci ikili sayı sistemine göre çalışan, komut dizilerini işleyen, aritmetiksel ve mantıksal işlemleri yapan ve bunları denetleyen sistemdir. Mikrodenetleyici ise giriş ve çıkış birimleri düzenleyen, programlayan ve bu devreleri içinde bulunduran mikroişlemcilere denilir. Bu bilgiler ışığında mikrodenetleyici ile mikroişlemci arasındaki farkı belirlemek için her ikisinin de donanımını, kullanıldıkları uygulama alanlarını ve komut kümelerini incelemek gerekir.

Şekil 2.1'de mikrodenetleyicinin ayrıntılı blok şeması gösterilmişti. Mikroişlemci, şekilden de görüldüğü üzere kontrol birimi, aritmetik/mantıksal birim ve saklayıcı gruplarının tek bir kılıf altında toplanması ile oluşmaktadır. Mikrodenetleyici ise bir adet mikroişlemciye ek olarak dahili veri ve kod bellek, zamanlayıcı/sayıcı, kesme denetçisi, seri/paralel iletişim arabirimi, adres yolu, veri yolu ve denetim vb. ek çevre birimlerin tek bir kılıf altında toplanması ile oluşmaktadır. Kısaca mikrodenetleyici bir mikroişlemci ile birlikte çevre birimlerin tek bir entegre haline getirilmiş halidir.

Mikrodenetleyiciye dayalı bir sistem mikroişlemci temelli sistemlerden oldukça farklıdır. Mikroişlemcili bir sistem, büyük çapta ve çeşitli kullanıcı programlarını yürütmek, çok sayıda veri işlemek, duyarlı sayısal hesaplamaları gerçekleştirmek için kullanıldığında yüksek verim sağlayabilmektedir. Mikrodenetleyici tabanlı bir sistem ise yeniden programlama gerektirmeyen sürekli sabit bir programın yürütüldüğü durumlarda çok kullanışlıdır.

Kullanım amaçları farklı olduğu için komut kümeleri de farklıdır. Mikroişlemci daha karmaşık uygulamalarda kullanılacağı için hız ve kelime uzunluğu büyük olarak tasarlanmıştır. Bunun sonucu olarak büyük veri gruplarını işleyebilecek komutlara sahiptir. Fakat mikrodenetleyicilerde bit işlem yapan komutlar daha önemlidir.

2.2. MSC-51 Ailesi Mikrodenetleyicinin Tarihçesi

Intel firması tarafından 1976 yılında piyasaya sürülen 8048 mikrodenetleyicisi dünyada üretilen ilk mikrodenetleyicidir. Üretiminde yaklaşık 17.000 transistor kullanılan 8048, kısa sürede kontrol uygulamalarının değişmez elemanı olmuştur. Artan gereksinimleri karşılamak üzere Intel firması, 1980 yılında MCS-51 mikrodenetleyici ailesinin ilk ürünü olan 8051 mikrodenetleyicisini piyasaya sürmüştür. 60.000 transistor içeren 8051 mikrodenetleyicisi Intel'den üretim izni alan birçok firma tarafından günümüzün ihtiyaçlarına cevap verecek şekilde gün geçtikçe geliştirilmekte ve yeni teknolojiler içermektedir.

Artık bir endüstri standardı haline gelen 8051 mikrodenetleyicisinin, istenen uygulamalara yönelik çok geniş ürün yelpazesi sunması, ucuz ve kolay temin edilebilmesi, birçok firma tarafından üretilmesi ve desteklenmesi, kitap/uygulama notu ve İnternet dokümanlarının rahatlıkla temin edilebilmesi vb üstünlükleri nedeniyle kullanımı gün geçtikçe yaygınlaşmaktadır.

Tablo 2. 1. 8051 üreten firmalar ve ürettikleri modellerin bazıları[1]

	Model	Veri Belleği		Kod Belleği			Haberleşme Protokolü				Z/S	WD	ADC	Port
		RAM	XRAM	ROM	EEPROM	FLASH	UART	I2C	CAN	SPI				
ATMEL	T80C51	128	-	4K	-	-	Var	-	-	-	2	-	-	32
	T83C51RB2	256	256	16K	-	-	Var	-	-	-	3	Var	-	32
	T89C51RC2	256	1K	-	-	32K	Var	-	-	Var	3	Var	-	48
	AT89S4D12	256	-	-	-	132K	Var	-	-	Var	3	-	-	40
	T89C51CC01	256	1K	-	2K	32K	Var	-	Var	Var	3	Var	10-bit	53
INTEL	80C31	128	-	–	-	-	Var	-	-	-	3	-	-	32
	80/87C51	128	-	4K	-	-	Var	-	-	-	3	-	-	32
	80C52	128	-	8K	-	-	Var	-	-	-	3	-	-	32
PHILIPS	80C528	256	256	-	-	-	Var	-	-	-	3	Var	-	48
	80C557	256	1792	-	-	-	Var	-	-	-	3	-	10-bit	40
	87C591	256	256	-	16K	-	Var	Var	Var	-	3	Var	10-bit	32
	89C668	256	8K	-	-	64K	Var	Var	-	Var	3	-	-	40
	8xC51RD2	256	768	-	-	64K	Var	-	-	Var	3	Var	-	32
DALLAS	DS5000(T)	128	32K	-	-	-	Var	-	-	-	2	-	-	32
	DS5002(FP)	128	128K	-	-	-	Var	-	-	-	2	-	-	32
	DS83C520	256	1K	16K	-	-	Var	-	-	-	3	Var	-	32
	DS80C390	256	4K	-	-	-	Var	-	-	-	3	Var	-	32
	DS89C420	256	1K	-	-	16K	Var	-	-	-	3	Var	-	32
Cygnal	C8051F005	256	2K	-	-	32k	Var	Var	-	Var	4	-	12-bit	64
	C8051F020	256	4K	-	-	64K	Var	Var	-	Var	5	-	12-bit	64
	C8051F300	256	-	-	-	8K	Var	-	-	Var	3	-	8-bit	32

Tablo 2.1'de bazı büyük tümdevre üretici firmaların üretmekte olduğu 8051 tabanlı mikrodenetleyicilerin temel özellikleri görülmektedir.

2.3. 8051'in Genel Yapısı ve Çalışma Prensibi

8051 mikrodenetleyicileri çalışmaya ilk başlatıldığında Program Sayacı (PC) ilk değer olarak 0000H adresindeki koda işaret eder. Bu adresteki opcode (komut kodu) alınarak program çalıştırılmaya başlanır. Kodu alınan komut çözülüp işletildikten sonra PC bir sonraki komutu işaret eder. PC'nin değerinin değişmesine sebep olacak herhangi bir komut yürütülmediği sürece de sıralı bir şekilde aynı işlemler yapılamaya devam edecektir. PC'nin değişimi sağlayan komutlar dallanma, çağırma ve kesme komutlarıdır.

Dallanma komutları bir koşul yerine geldiğinde gerçekleşebileceği gibi doğrudan başka bir kod bölümüne atlama şeklinde de gerçekleşebilir. Bu durumda programın çalışması o noktadan itibaren devam edecektir ve normal çalışma sürecinde geri dönülmeyecektir.

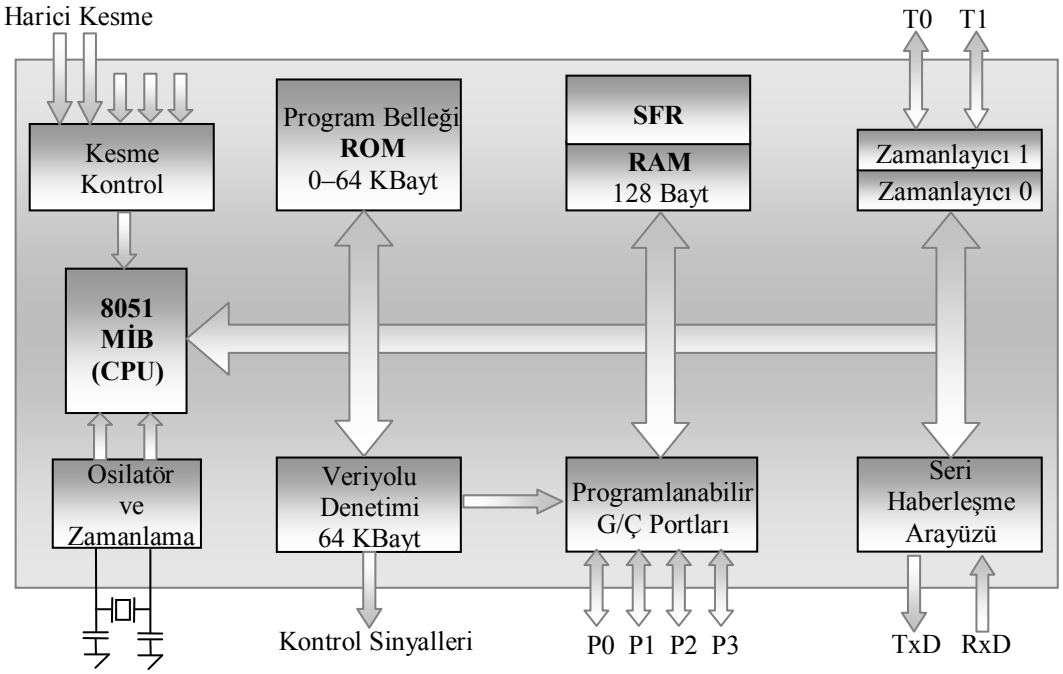
Ayrıca alt programlara erişim alt programları çağırma şeklinde de gerçekleştirilebilir. Dallanmalardan farklı olarak alt programlar çalışmaları bittikten sonra RET komutu ile çalışma sürecini kendilerini çağıran programa iade ederler. [12]

Kesmelerde ise durum daha farklıdır. Kesmede işlenen kesme hizmet programı, bir alt program olmasına rağmen programcıdan bağımsız olarak fiziksel bir koşul gerçekleştiğinde mikroişlemci otomatik olarak bu alt programı işlemeye başlar. Alt programlardan farklı olarak RETI komutu ile dönerler. [12]

Günümüzde 8051 tabanlı mikrodenetleyiciler daha düşük güç tüketen CMOS teknolojisi ile üretilmektedir ve 80C51 şeklinde adlandırılırlar. 8051 mikrodenetleyicisinin standart özellikleri şunlardır:

- Kontrol uygulamalarına yönelik 8 bit CPU
- Mantıksal işlemci (tek-bit lojik işlemler)
- 64 KB program hafıza ve veri hafıza adres alanı

– 4K ROM, (0-64K arasında)
– 128 Bayt RAM, (256 bayt'a çıkabilir)
– 4 tane 8-bit Giriş/Çıkış portu (32 uç)
– 2 tane 16-bit zamanlayıcı/sayıcı
– Full duplex UART (Universal Asynchronous Receiver Transmitter)
– İki öncelik seviyesine sahip 6-kaynak/5 vektörlü kesme donanım yapısı



Şekil 2.2. 8051 Blok Diyagramı[1]

2.3.1. İşlemci Çekirdeği (CPU)

Her mikrodenetleyici bir mikroişlemci çekirdeği bulundurmak zorundadır ve bu birim Merkezi İşlem Birimi (MİB) olarak da adlandırılır. MİB, ROM bellekte bulunan programın çalıştırılmasından sorumludur ve üç ana bloktan oluşmaktadır: ALU, program sayıcıları, ve çeşitli saklayıcılar.

CPU (Çekirdek):

- Verileri okur veya depolar
- Basit aritmetik işlemleri yapar
- Mantıksal işlemleri (VE, VEYA gibi) gerçekleştirir

- Programın akışını denetler.

Üreticiler arasında işlemci çekirdeklerinin farkları:

- Mimari tipleri: Harvard, Von Neuman, Hibrid
- Sözcük genişliği: 4, 8, 16, 32, 64 bit
- Komut kümeleri: RISC, CISC
- Saklayıcı türü ve adetleri
- Adresleme metotları
- Kesme adetleri
- Hız/güç/boyut.

2.3.2. Sistem Belleği

Mikrodenetleyiciler bellek birimlerini üzerinde bulundururlar ve bunların türleri ve büyüklükleri aileden aileye değişiklik gösterir. Mikrodenetleyicinin mutlaka bir dahili (on-chip) ROM belleğe sahip olması gerekmeyebilir (8031 'de olduğu gibi). Ancak her mikrodenetleyici büyüklüğü farklı da olsa bir RAM bellek bulundurmak zorundadır. Genellikle en az 128 Byte'lık bir RAM bellek üreticiler tarafından standart hale getirilmiştir. Buna ek olarak Özel Fonksiyon Saklayıcıları (SFR-Special Function Registers) isimli bir RAM bellek de bulunmaktadır. Böylelikle 256 Byte'lık bir RAM bellek 8051 mikrodenetleyicileri için minimum bellek büyüklüğü olarak kabul edilebilir.

Program Belleği (Kod Belleği): Mikrodenetleyici sistemlerin çalışabilmesi için gerekli olan bellek ROM bellektir. Bu bellekte, çalışacak programın makine kodları bulunmaktadır. Program depolama için kullanılan ROM bellekler farklı teknolojilerde üretilmektedirler.

Veri Belleği (RAM): RAM bellekler ROM belleklerin aksine kendilerini besleyen güç kesildiğinde muhafaza ettikleri bilgileri yitirirler. Bu bellekler sistem çalışırken içerisindeki bilgilerin sürekli değişimine izin verir yapıdadırlar. Geçici bellek alanı olarak kullanılırlar ve binlerce kez yazılıp okunabilirler. RAM bellekler genellikle

Statik RAM (SRAM) ve Dinamik RAM (DRAM) olmak üzere iki türde üretilirler. DRAM bellek tümdevreleri daha az yer kaplar ancak daha yavaş çalışırlar. Bunun nedeni muhafaza ettikleri bilgilerin sürekliliğini sağlamak için belirli aralıklarla tazeleme işaretlerine gerek duymaları ve bu esnada okuma yazma yapamamalarıdır.

SRAM bellekleri ise tazeleme işaretlerine ihtiyaç duymadan bilgiyi saklayabilirler. SRAM bellekler DRAM belleklere göre daha hızlı çalışır fakat tümdevre olarak daha fazla yer kaplarlar. Ayrıca SRAM belleklere erişim arayüzü oldukça basit olup ek birimlere ihtiyaç duyulmaz. Bu yüzden mikrodenetleyicilerde bellek ihtiyacı kilo byte'lar mertebesinde olduğundan genellikle SRAM türü bellekler kullanılır. Ancak örneğin kişisel bilgisayarlarda bu belleklerin pahalı olmasından ve mega byte'lar mertebesinde belleklere ihtiyaç duyulduğundan dolayı DRAM bellekler tercih edilir.

2.3.3. Çevresel Birimler (Peripherals)

8051 mikrodenetleyicisi standart birimlerin (RAM, ROM, ALU) yanında uygulamalarda kullanılacak ve harici sistemlerle etkileşimi sağlayacak bazı çevresel birimler içermektedir. Bunların sayısı ve özellikleri üreticiden üreticiye ve üretim serisine bağlı olarak çeşitlilik arz edebilir. Çevresel birimler olarak sıklıkla saat darbesi üreteçleri, zamanlayıcı/sayıcılar, A/D dönüştürücüler veya D/A dönüştürücüler, haberleşme birimleri, LCD sürücü üniteleri karşımıza çıkmaktadır. Tasarımcılar için gerekli kriterleri sağlayan mikrodenetleyici ailesini ve bu ailenin ilgili seri numaralı denetleyicisini seçmek çok önemli bir basamaktır. Bu seçimde, sadece karşılanması gereken donanım kriterleri değil başarım (performans) ve maliyet faktörleri de göz önünde bulundurulmalıdır.

Zamanlayıcı/Sayıcı Birimleri: Zamanlayıcılar (timer), olaylar arasındaki süreyi ölçmek için veya belirli aralıklarla işaretler üretmek amacıyla kullanılırlar.

Haberleşme Birimi: Mikrodenetleyicilerde harici ortam ile iletişimde hem seri hem de paralel iletişim yöntemleri kullanılabilir. Mikrodenetleyicilerin büyük bir kısmında UART seri haberleşme ünitesinin yanı sıra I^2C (Inter-Integrated Circuit), SPI (Serial Peripheral Interface) veya CAN (Controller Area Network) gibi seri haber-

leşme birimleri dâhili olarak bulunabilir. Seri iletişimde bilgiler, bit bit olarak ve belirli bir protokol dâhilinde gönderilir. USB, CAN, UART ve burada sayılamayan birçok seri iletişim protokolleri endüstride olduğu gibi mikrodenetleyici bünyelerinde de yoğun bir şekilde kullanılmaktadır.

UART, birçok çeşit seri iletişim protokolüne cevap verebilecek bir seri çevresel birimdir. Mikrodenetleyici işlemcisinden gelen paralel bilgileri seriye ve dış ortamdan gelen seri bilgileri de paralel forma çevirir.

Giriş/Çıkış Portları: Mikrodenetleyici, üzerinde bulunan G/Ç portları aracılığıyla da harici ortam ile haberleşir. Ancak bu haberleşme seri haberleşmenin aksine bir protokol dahilinde gerçekleşmez. Porttaki tüm uçlar tamamen birbirinden bağımsız ve asenkron olarak harici sistemlere işaret gönderir veya alırlar. Portlar anahtar takımlarını okumak ya da bellek entegrelerine adres ve veri yolu olmak gibi daha birçok farklı uygulamada kullanılabilirler. Paralel G/Ç bilgiyi gruplar halinde alır ve verir. Bu tür bir iletişim, röle veya anahtar değerlerinin değiştirilmesi gibi bilgilerin sık değişmediği durumlarda daha verimlidir.

Kesme Kontrol Birimi: Mikrodenetleyicinin mevcut çalışmasını sürdürürken gelen bir kesme işareti ile o an yapmakta olduğu işi bırakıp, kesme işaretinin gerektirdiği işe dallanması ve bu işi bitirdikten sonra tekrar bıraktığı işe devam etmesini sağlayan birimdir. Kesme birimi ile mikrodenetleyici çevre birimlerden gelen isteklere gerçek zamanlı olarak yanıt verebilir.

Saat Darbesi Üreteci: Dahili saat devresi, mikrodenetleyici içerisindeki işaretlerin zamanlamasını denetlemek ve düzenlemek üzere bir kare dalga işaret üretir. Saat devresi, tümdevre içerisindeki işlemlerin adım adım yürütülmesi için osilatör devresinin ana parçası olan kristal elemanını kullanır. Kristal aynı zamanda mikrodenetleyicinin çalışma frekansını belirler. Osilatör devresi devamlı dalga formunda bir elektriksel işaret üretir ve bu işaret saat devresi tarafından çoğunlukla bir PLL (Phase Locked Loop, Faz Kilitlemeli Döngü) birimi yardımıyla kararlı kare dalga biçimine çevrilir. 30 MHz'de çalışan bir mikroişlemci veya mikrodenetleyici ile karşılaşıldığında, bu entegrelere ait saat devresinin bir saniyede 30 milyon adet kare

dalga şeklinde saat darbeleri ürettiği anlaşılmalıdır. Saat devresi, frekans çarpıcı veya bölücü de içerebileceği için tek başına kristal frekansının bilinmesi işlemci hızının belirlenmesinde yanıltıcı olabilir.

2.4. Assembly Dili ve Özellikleri

Mikrodenetleyicilere istenilen bir işlemi icra ettirmek için bu işlemi, anlayabilecekleri şekilde kodlarla ifade etmek gerekir. Mikrodenetleyiciler ve mikroişlemciler sadece '1' ve '0'lardan oluşan ve komut (opkod) olarak adlandırılan mantıksal bit dizilerini yorumlayabilirler. Bitler seviyesinde yapılan tasarımların kolay anlaşılamaması, gerçekleştirilen tasarımlar üzerinde değişiklik yapabilmenin zorluğu gibi sebeplerden dolayı kullanıcıların kolaylıkla anlayabileceği şekilde uygulama geliştirmesine imkan tanıyan programlama dilleri geliştirilmiştir. Ama kullandığınız programlama dili ne olursa olsun, bir simgesel dildir ve mikroişlemcilerin icra edebileceği bir forma dönüştürülmek zorundadır.

Assembly dilinde her bir komut, gerçekleştirdiği işleve karşılık gelen İngilizce sözcüğün kısaltması (mnemonik) ile ifade edilir. Her bir kısaltma, makine dilindeki farklı bit dizisine karşılık gelmektedir.

Örneğin:

MOV A, #55 ;Akümülatöre 55_{10} değerini yükle

Yukarıdaki komut satırında MOV kısaltması bir verinin belirtilen yere transferi işlevini yerine getirmektedir.

Assembly dilini kullanarak uygulama geliştirmek makine diline göre daha kolayken C, BASIC, PASCAL vb gibi yüksek düzeyli dillerde program yazmaya göre daha zordur. Çünkü assembly dilinde program yazmak için, programın kullanılacağı donanım, adresleme yöntemleri, komut kümesi hakkında ayrıntılı bilgi sahibi olmak gerekmektedir. Bunun yanında yüksek düzeyli dillerde ise bu gibi teknik ayrıntıyı bilmeksizin uygulama geliştirmek mümkündür.

Kullanım açısından assembly gibi düşük düzeyli dillerin yüksek düzeyli dillere göre birkaç dezavantajının olmasının yanında birçok önemli avantajları da mevcuttur. Örneğin; donanım hakkında daha fazla detay bilmeyi gerektirir. Bu dezavantaj gibi görülse de aslında kullanıcıya önemli bilgi birikimi sağlamaktadır. Özel donanım ihtiyaçları üzerinde daha fazla kontrol sağlar. Yüksek seviyeli dillere göre daha küçük, daha az yer kaplayan ve daha hızlı icra edilebilir kodlar üretilebilir.

2.4.1. Program Formatı

Assembly dilinde program geliştirmek için bu dilin genel kodlama formatının, açıklama satırlarının, simge ve veri tanımlamalarının bilinmesi gereklidir. Aşağıda assembly dilinde yazılmış bir program parçasındaki kısımlar görülmektedir. Verilen örnek içerisindeki tüm kısımları tek tek inceleyelim.

	İşlem Kodu		
Etiket	(Komut–Opkod)	İşlenen (Operand)	Açıklama
Basla:	MOV	A, #77h	;Aküye 77_{16} değerini yükle

Etiket alanı: Komut satırının ilk bilgisidir ve sembolik isimlerden oluşur. Program içerisindeki belirli işlevlerin gerçekleştiği bölümlerin başlangıcını göstermek amacı ile kullanılır. Program içersinde istenilen kısma kolaylıkla dallanılmasını sağlar. Etiket ismi olarak mikroişlemci komut setinde tanımlı olan bir komut ismi verilemez. Etiket isimleri bir harf ile başlamak zorundadır.

Komut: Kısaltma(mnemonik) olarak da adlandırılan, komut seti içerisinde mikroişlemcinin belirli bir işi yapmasını sağlayan tanımlanmış sembollerdir. Komut alanına etiketten sonra 1 boşluk ya da sekme (tab) ile girilir.

İşlenen: Bu alan, işlemciye işlenecek veriyi ya da verinin nerede olduğunu gösterir. Tek başına bir anlam ifade etmez. Genelde komutun etki edeceği hedef ve kaynak bilgisini içerir. Hedef ve kaynak bilgisi birbirinden virgül (,) ile ayrılır.
İşlenen (operand) kısmında işlenecek bilgi farklı sayı sistemlerinde ifade edilebilir. Kullanılan sayının hangi sayı sistemine ait olduğunu bilginin önüne ya da sonuna

konulan özel işaretler belirler. Tablo-2.2'de 4 farklı sayı siteminin assembly dilindeki kullanımı görülmektedir.

Tablo 2. 2. Assembly dilinde sayı sistemlerinin kullanımı [1]

Ön Takı	Son Takı	Anlamı	Örnek
(Boşluk)	D	Onlu sayı (decimal)*	55 – 55D
%	B	İkili sayı (binary)	%01010101 – 01010101B
@	O	Sekizli sayı (octal)	@33 – 33O
$	H	Onaltılık sayı (hexadecimal)	$FB – FBH

Açıklama Satırı: Assembly dili (;) ile başlayan satırları açıklama satırı olarak kabul eder. Bu satırları yorumlamaz ve makine kodu üretmez. Yazılan uygulamanın anlaşılırlığını arttırır.

2.4.2. Yönergeler (Directives)

Assembly dilinde kullanılan birçok yönerge (talimat) mevcuttur.

ORG: Kod bellek içerisinde programın başlangıç adresini belirtmek için kullanılan adres konumlandırma talimatıdır. Bu talimat **ORG 'Adres'** şeklinde kullanılır. Programlar genelde belleğin başlangıç adresi olan 0000h'dan başlar. ORG talimatındaki adres değeri program sayıcıya (PC) yüklenerek programın yazılacağı adres belirlenir. Bir program içerisinde birden fazla ORG komutu kullanılabilir.

```
Talimat         Açıklama
ORG 0000h       ;program 0000h adresinden başlasın PC=0000
ORG 0030h       ;program 0030h adresinden başlasın PC=0030
```

END: Programın bitiğini gösteren talimattır.

DB (Define Bayt): Kod bellek içerisinde sayı ve kelime (string) dizilerinin tanımlanmasını sağlar.

```
İsim            DB    İfadeler                Açıklama
```

Max_sayi	DB	255	;tek bir değişken tanımlanması
Tablo	DB	0, 5, 4, 3, -10	;dizi olarak tanımlama
Yaz	DB	‘8051 Ogreniyorum’	;string olarak tanımlama

EQU: EQU (Equal = eşittir) bir sayısal değerin istenilen sembol adına atanması işlemini gerçekleştirir. Bu tanımlama program içerisinde bir ifadenin ya da değerin çok fazla tekrar edildiğinde programın anlaşılırlığını arttırmak için kullanılır.

İsim	Talimat	Değer	Açıklama
Pi	EQU	3.14	;sabit değer tanımlama
Bilgi	EQU	55h	;55h adresindeki veriyi bilgi değişkenine ata

2.5. Adresleme Yöntemleri

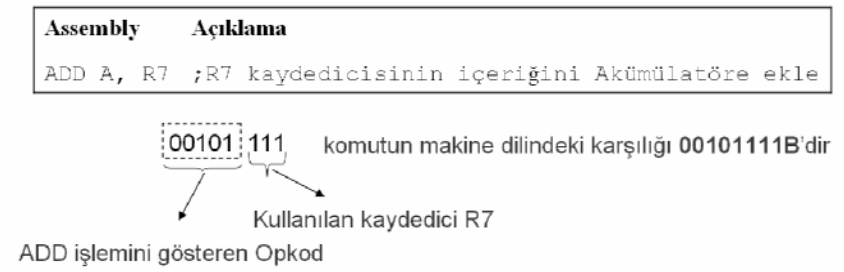
Adresleme modu, bir bellek konumuna ya da bir veriye erişimin nasıl olacağını belirtir. Doğrudan kullanılan komut uzunluğunu etkiler. Kullanılan komutlara bağlı olarak bilginin farklı yollarla hedefe gitmesine olanak sağlar. 8051 mikrodenetleyicisinde kullanılan 8 farklı adresleme yöntemi şunlardır:

- Kaydedici adresleme
- Doğrudan adresleme
- Dolaylı adresleme
- İvedi adresleme
- Bağıl (Koşullu) adresleme
- Mutlak adresleme
- Uzun adresleme
- İndisli adresleme yöntemi

2.5.1. Kaydedici Adresleme

8051 mikrodenetleyicisinde R0'dan R7'ye kadar 8 tane genel amaçlı kaydedici vardır. Kaydedici adreslemede komutu oluşturan en yüksek değerlikli 5 bit yapılacak işlevi ve en düşük değerlikli 3 bit ise R0 ile R7 arasındaki hangi kaydedicinin kullanılacağını gösterir.

Şekil 2. 3.Saklayıcı adresleme



2.5.2. Doğrudan Adresleme

Doğrudan adresleme yöntemi, dahili alt RAM (lower RAM) ve SFR alanına erişmek için kullanılır. Doğrudan adresleme yönteminde komutlar 2 bayt uzunluğundadır. İlk bayt opkod'u (gerçekleştirilecek işlemi), ikinci bayt adres bilgisini gösterir. Doğrudan adresleme yöntemi adresleri örtüşen ÜST RAM ile SFR bölgeleri birbirinden ayrılmasını sağlar. Bu iki alandan SFR bölgesine doğrudan adresleme yöntemi kullanılarak erişilebilir.

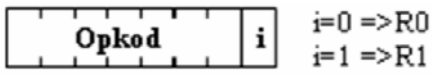
Şekil 2. 4. Doğrudan adresleme

Assembly	Açıklama
MOV P1,A	;Aküyü Port 1'e kopyala
MOV A,70h	;70h adresinin içeriğini Aküye kopyala
MOV A,80h	;SFR bölgesine erişilir, 80h Port 0'ın adresidir. P0'daki bilgi Aküye kopyala

2.5.3. Dolaylı Adresleme

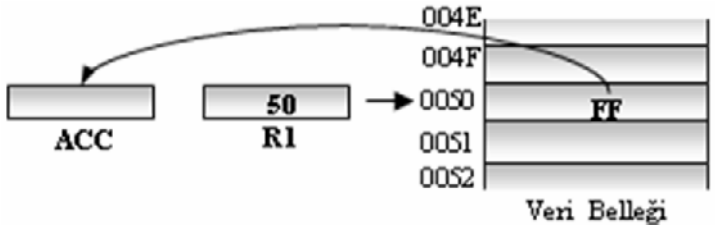
Bir programın çalışması esnasında tanımlanan bir değişkenin adresinin değiştirilmesi, hesaplanması ya da tekrar değiştirilmesi işlemlerinde dolaylı adresleme yöntemi kullanılmaktadır. Dolaylı adresleme, adresleme yöntemlerinin en güçlüsüdür.

Bu adreslemede kaynak veya hedefin adresi komutun içerisinde açık olarak verilmez. Verinin gerçek adresini tutmak için R0 ve R1 saklayıcıları "işaretçi" olarak kullanılır. Bu saklayıcılar bilginin RAM'de yazılacağı veya okunacağı adresi içermektedirler. Opkodun en küçük değerlikli biti (LSB) kullanılacak olan saklayıcıyı (R0 veya R1) belirler. R0 ve R1 saklayıcıları 8 bitlik olduğundan 256 Byte'lık bir bellek alanı adreslenebilmektedir. Ancak ÜST RAM ile örtüşen adrese sahip olan SFR'ye kesinlikle erişilemez.



Şekil 2. 5. Dolaylı adresleme

8051 assembly dilinde dolaylı adresleme, R0 veya R1 saklayıcılarının önüne "at" @ işareti getirilerek gerçekleştirilir. Bu adresleme yöntemi veri transfer işlemlerinde kullanılır. Aşağıdaki örnekte görüldüğü gibi dolaylı adreslemeyi kullanarak alt (lower) RAM'deki 50h adres bölgesinin içeriğini akümülatöre kopyalama işleminin nasıl olacağı gösterilmektedir.



Şekil 2. 6. Dolaylı adresleme örneği

Assembly	Açıklama
MOV A,@R1	;Alt RAM'deki 50h adresinin içeri (FFh) Aküye aktar

2.5.4. İvedi Adresleme

Bir assembly programının derlenme aşamasında bilinen bir sabit değerin kullanılabilmesi için değerin komut içerisine koyulması en hızlı çözümü üretir. Bir diğer önemli prensip hedefe veriyi aktarmanın en kolay yolu opkod'u kaynak yapmaktır. İvedi adresleme bu iki prensibin gerçeklenmesini sağlayan bir adresleme yöntemidir. İvedi adresleme DPTR'nin kullanıldığı istisnai durum dışında 2 Byte uzunluğundaki komutlardan oluşur. Bilginin geçici olarak komut içerisine yüklenmesi yüksek komut hızı sağlar. Bilginin komut içerisinde yer alması, sabit derleme zamanı sağlamaktadır. Böylelikle komutların yaklaşık çalışma süreleri hesaplanabilir.

Şekil 2. 7. İvedi adresleme

Assembly dilinde ivedi adresleme yöntemi, (#) işaretinin sayıdan önce kullanılması ile gerçekleştirilir.

Assembly	Açıklama
MOV A,#12	;Akü'ye 12 değerinin atılması
MOV R0,#10h	;10h bilgisini R0 saklayıcısma yükle
MOV DPTR,#2000H	;2000h bilgisini DPTR'ye yükle, 3 Byte'lık komut

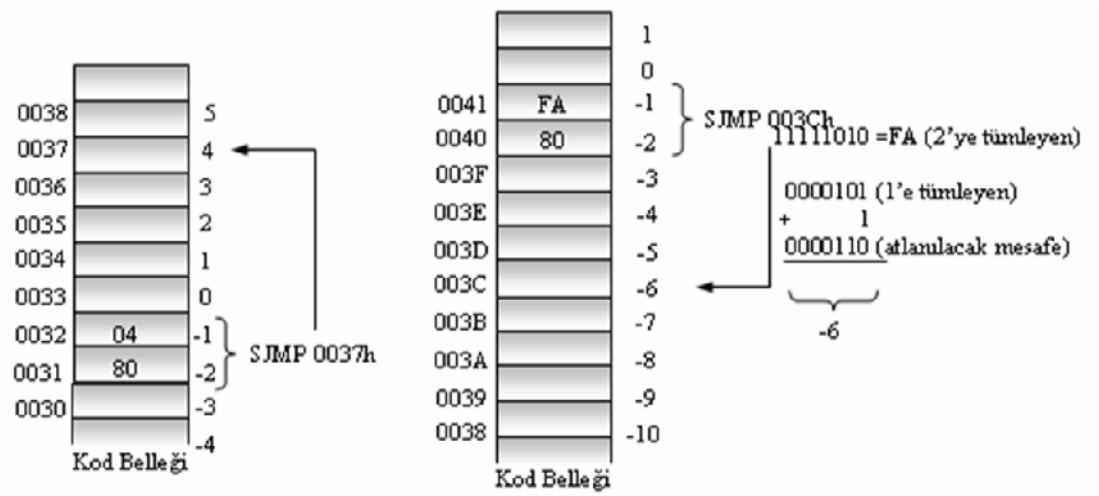
2.5.5. Bağıl Adresleme

Bağıl adresleme yöntemi sadece atlama komutları ile birlikte kullanılır. Bu yöntemde komutlar 1 Byte opkod ve 1 Byte adres bilgisi olmak üzere toplam 2 Byte uzunluğundadır. Adres bilgisi 8-bit ile ifade edildiği için maksimum (2^8) 256 Byte'lık bir alanda +127 (ileri yön) ve -128 (geri yön) aralığında bir atlama işlemi gerçekleştirilir.

Şekil 2. 8. Bağıl adresleme

Atlama (80h) komutunun ardından program sayıcının (PC) adresi bir arttırılır. Böylelikle yeni adres, atlama komut adresine değil sonraki komuta bağlıdır. Adres bilgisi-

nin 1. biti ‘1’ ise eksi kabul edilir ve atlama işlemi geriye doğru gerçekleştirilir. Eğer 1. bit ‘0’ ise artı kabul edilir ve ileriye doğru atlama işlemi gerçekleştirilir.



Şekil 2. 9. Bağıl adresleme yöntemi

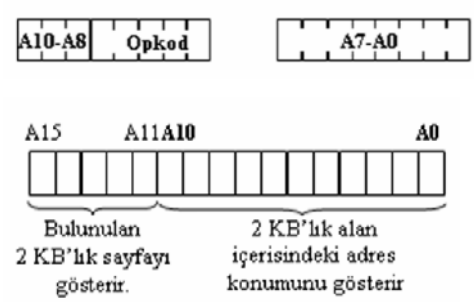
Assembly		**Açıklama**
	MOV A,#FFh	;Akü'ye FF_{16} değerinin atılması
Git:	MOV Pl,A	;Akü'yü Pl portuna yükle
	DEC A	;Akü'yü bir azalt
	SJMP Git	;Git etiketine dallan

Yukarıda bir bağıl adresleme örneği görülmektedir. Koşulsuz dallanma komutu (sjmp) kullanıldığından program sonsuz döngü şeklinde FFh değerini sürekli bir azaltarak Pl portuna gönderir.

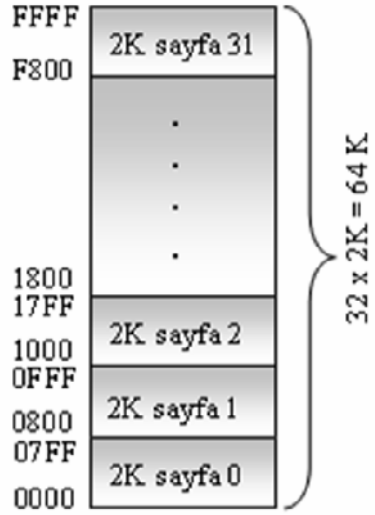
2.5.6. Mutlak Adresleme

Mutlak adresleme yöntemi sadece **ACALL** ve **AJMP** komutları ile kullanılır. Bu komutlar 2 Byte uzunluğundadır ve kod bellek içerisinde 2 Kbyte'lık bir alanı adresleyebilirler. Maksimum 64 KByte olan kod bellek aşağıdaki şekilde görüldüğü gibi 2 Kbyte'lık 32 bölmeye ayrılabileceğinden hangi bölmenin seçileceğini program sayacı (PC) belirlemektedir.

Şekilde görüldüğü gibi hedef adresin yüksek değerlikli 5 biti (A15-A11), program sayıcının (PC) yüksek değerlikli 5 bitidir. Böylece A15-A11 arasındaki bitler değiştirilmediğinden sadece 2 Kbyte'lık sayfa içerisinde atlama yapılabilir.



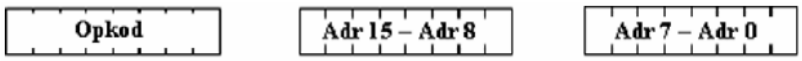
Şekil 2.10. Mutlak adresleme



Şekil 2.11. 64 kB'lık belleğin sayfalara ayrılması

2.5.7. Uzun Adresleme

Uzun adresleme yönteminde yalnızca 3 Byte'lık **LCALL** ve **LJMP** komutları kullanılır. Bu komutlarda 16-bit hedef adres bulunabilir. Bu adresleme yöntemi ile 2^{16} = 64 Kbyte'lık adres aralığında atlama işlemi gerçekleştirilebilir.

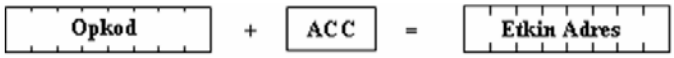


Şekil 2.12. Uzun adresleme

Modern assembly derleyicilerinde koşulsuz dallanma işlemi **JMP**, **SJMP**, **LJMP** komutları ile gerçekleştirilebileceği gibi sadece **JMP** komutu ile de gerçekleştirilebilir. Assembler komutunu, adresler arasındaki mesafeye göre **SJMP** ya da **LJMP** olarak belirleyebilir. Bu işlem tasarımcı için büyük kolaylık sağlar.

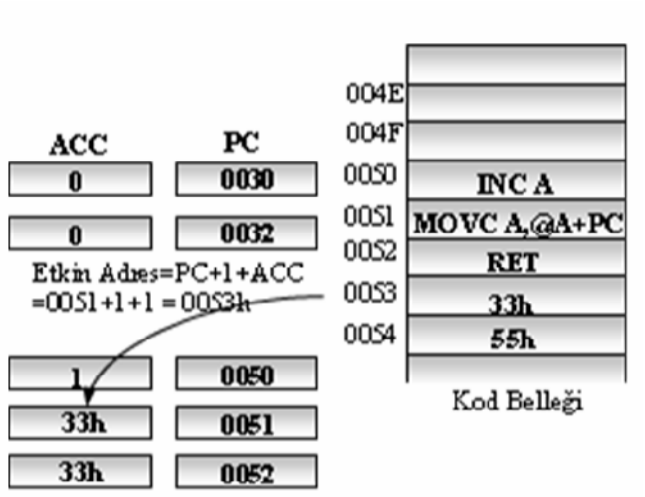
2.5.8. Sıralı (index) Adresleme

Sıralı adresleme, bellekte bulunan sıralı bilgilere erişmek ve tablo (look-up table) halindeki sabit verileri kullanmak için en elverişli adresleme yöntemidir. Bu yöntemde çok sayıda veriye az sayıda komut kullanarak erişmek olasıdır. Böylelikle bellek alanından tasarruf sağlanır. Sıralı adresleme yönteminde, **JMP** ve **MOVC** komutları kullanılır. Bu yöntemde program sayacı (PC) veya veri işaretçisi (DPTR) ile akümülatörün toplamı, atlanılacak olan etkin adres bilgisini belirler.



Şekil 2. 13. Sıralı adresleme

```
Adres     Kodlar
0030h             MOV A,#0
0032h             CALL BASLA
                  -----
                  -----
0050h     BASLA:  INC  A
0051h             MOVC A, @A+PC
0052h             RET
0053h     TABLO:  DB 33h
0054h             DB 55h
                  -----
```



Şekil 2. 14. Sıralı adresleme örneği

Yukarıdaki şekilde sıralı adresleme yönteminin kullanıldığı bir örnek görülmektedir. Program, aküye 0h değerinin atılması ile başlamaktadır, **CALL BASLA** komutu ile 50h adresinde bulunan alt programa dallanılır, **BASLA** alt programı her çağrıldığında tablo halindeki veriler sıra ile alınmaktadır. Tablodaki verilerin sıra ile alınmasını sağlayan akünün değeri **INC A** komutu ile 1 arttırılır. **MOVC A, @A+PC** komutu çalışması gereği önce PC'nin o anda gösterdiği değeri 1 arttırır, sonra akü ile toplayıp tablodan alınacak verinin yerini gösteren etkin adresi (0053h) elde eder ve bu adresteki veriyi (33h) alarak aküye yükler.

2.6. 8051 Komut Türleri

8051'de de kullanılan komutlar 8-bit opkoda sahiptir. 8-bit opkod 2^8=256 farklı komuta imkân tanır ve 8051'de toplam 255 komut tanımlıdır. 8051 komut kümesi 1, 2 ya da 3 bayt uzunluğunda komutlardan meydana gelmektedir. Komut kümesini oluşturan 255 komutun 139'u 1 bayt, 92'si 2 bayt ve 24'ü 3 bayttır. 8051 komut kümesi beş ana başlık altında incelenebilir:

- Aritmetik komutlar
- Mantıksal komutlar

– Veri transfer komutları
– Bit işlem komutları
– Program dallanma komutları

2.6.1. Aritmetik Komutlar

Mikrodenetleyici uygulamaları, çoğunlukla program içerisindeki eylemleri ve program akışını değiştirmek amacıyla veri üzerinde matematiksel işlemlerin yapılmasını gerektirir. Uygulamalarda özellikle istenen gerçek zamanlı kontrol işlemlerinin gerçekleştirilmesidir. Bu da mikrodenetleyici ile aritmetik işlemlerin etkin olarak yapılması ile mümkündür.

Aritmetik işlem komutları, toplama, çıkarma, arttırma, azaltma, çarpma, bölme ve onluk tabana uyarlama komutlarından oluşmaktadır. 8051 mikrodenetleyicisinde kullanılan tüm aritmetik komutlar, kullanıldıkları adresleme yöntemlerine, etkiledikleri bayrak türüne ve Byte olarak uzunluklarına göre aşağıdaki tabloda verilmiştir.

Tablo 2. 3.Aritmetik komut kümesi [1]

	Komut		Açıklama	Bayrak	Bayt
Toplama	ADD	A, Rn	Rn saklayıcı değerini akümülatöre ekle	C,OV,AC	1
	ADD	A, adres	Adresteki bilgiyi Aküye ekle	C,OV,AC	2
	ADD	A, @Ri	Saklayıcının gösterdiği adresteki bilgiyi aküye	C,OV,AC	1
	ADD	A, #bilgi	Doğrudan bilgiyi aküye ekle	C,OV,AC	2
Eldeli Toplama	ADDC	A, Rn	Akümülatör ile saklayıcı değerini elde ile topla	C,OV,AC	1
	ADDC	A, adres	Elde ile aküye adresteki bilgiyi ekle	C,OV,AC	2
	ADDC	A, @Ri	Elde ile saklayıcının gösterdiği adresteki bilgiyi aküye ekle	C,OV,AC	1
	ADDC	A, #bilgi	Elde ile doğrudan bilgiyi aküye ekle	C,OV,AC	2
Çıkarma	SUBB	A, Rn	Borç ile Aküden saklayıcının değerini çıkart	C,OV,AC	1
	SUBB	A, adres	Borç ile aküden adresteki bilgiyi çıkart	C,OV,AC	2
	SUBB	A, @Ri	Borç ile saklayıcının gösterdiği adresteki bilgiyi aküden çıkart	C,OV,AC	1
	SUBB	A, #bilgi	Borç ile Aküden bilgiyi çıkart	C,OV,AC	2

Arttırma	INC	A	Akümülatörü 1 arttır	–	1
	INC	Rn	Saklayıcıyı 1 arttır	–	1
	INC	adres	Adresteki bilgiyi 1 arttır	–	2
	INC	@Ri	Ri saklayıcının gösterdiği adresteki bilgiyi 1 arttır	–	1
	INC	DPTR	DPTR saklayıcısını 1 arttır	–	1
Azaltma	DEC	A	Akümülatörü 1 azalt	–	1
	DEC	Rn	Saklayıcıyı 1 azalt	–	1
	DEC	adres	Adresteki bilgiyi 1 azalt	–	2
	DEC	@Ri	Ri saklayıcısının gösterdiği adresteki bilgiyi 1	–	1
Çarpma	MUL	AB	A ve B saklayıcılarının içeriklerini çarp. Çarpım sonucunda yüksek değerlikli bayt B saklayıcısına, düşük değerlikli bayt ise Akü'ye	C,OV	1
Bölme	DIV	AB	A'yı B'ye böl. İşlem sonucunda Bölüm Akü'ye, Kalan B saklayıcısına yüklenir.	C,OV	1
	DA	A	Akümülatörü onluk tabana ayarla	C	1

2.6.2. Mantıksal Komutlar

Lojik işlem komutları VE, VEYA, Özel VEYA, sola ve sağa döndürme komutları ile akünün 4'lüklerinin (nibble) yerini değiştirme komutundan oluşmaktadır. 8051 mikrodenetleyicisinde kullanılan tüm mantıksal komutlar, kullanım şekillerine, etkiledikleri bayrak türüne ve Byte olarak uzunluklarına göre tabloda verilmektedir.
8051 mantıksal komutları verinin Byte'ları üzerinde bit tabanlı lojik işlem yapar. Akümülatörü kullanan tüm mantıksal komutlar 1 makine çevriminde çalışır. Diğer komutlar ise 2 veya 3 makine çevriminde çalışırlar.

Tablo 2. 4. Mantıksal işlem komutları [1]

	Komut		**Açıklama**	**Bayrak**	**Bayt**
VE işlemi	ANL	A, Rn	Rn saklayıcısı ile aküyü VE işlemine tabi tut	–	1
	ANL	A, adres	Adresteki bilgi ile aküyü VE işlemine tabi tut	–	2
	ANL	A, @Ri	Saklayıcının gösterdiği adresteki bilgi ile aküyü lojik VE işlemine tabi tut	–	1
	ANL	A, #bilgi	Doğrudan bilgi ile aküyü VE işlemine tabi tut	–	2
	ANL	adres, A	Akü ile adresteki bilgiyi VE işlemine tabi tut	–	2
	ANL	adres, #bilgi	Bilgi ile adresteki veriyi VE işlemine tabi tut	–	3
VEYA işlemi	ORL	A, Rn	Rn ile akümülatörü VEYA işlemine tabi tut	–	1
	ORL	A, adres	Adresteki bilgi ile aküyü VEYA işlemine tabi tut	–	2
	ORL	A, @Ri	Saklayıcının gösterdiği adresteki bilgi ile aküyü VEYA işlemine tabi tut	–	1
	ORL	A, #bilgi	Doğrudan bilgi ile aküyü VEYA işlemine tabi tut	–	2
	ORL	adres, A	Akü ile adresteki bilgiyi VEYA işlemine tabi tut	–	2
	ORL	adres, #bilgi	Bilgi ile adresteki veriyi VEYA işlemine tabi tut	–	3

Özel VEYA işlemi	XRL	A, Rn	Rn ile aküyü Özel VEYA işlemine tabi tut	–	1
	XRL	A, adres	Adresteki bilgi ile aküyü Özel VEYA işlemine tabi tut	–	2
	XRL	A, @Ri	Saklayıcının gösterdiği adresteki bilgi ile aküyü Özel VEYA işlemine tabi tut	–	1
	XRL	A, #bilgi	Doğrudan bilgi ile aküyü Özel VEYA işlemine tabi tut	–	2
	XRL	adres, A	Akü ile adresteki bilgiyi Özel VEYA işlemine tabi tut	–	2
	XRL	adres, #bilgi	Bilgi ile adresteki veriyi Özel VEYA işlemine tabi tut	–	3
Döndürme İşlemi	RL	A	Akümülatörü 1 bit sola döndür	–	1
	RR	A	Akümülatörü 1 bit sağa döndür	–	1
	RLC	A	Akümülatörü elde üzerinden 1 bit sola döndür	C	1
	RRC	A	Akümülatörü elde üzerinden 1 bit sağa döndür	C	1
	SWAP	A	Akümülatörün ilk dört biti ile son dört bitini yer değiştir	–	1

2.6.3. Veri Transfer Komutlar

Veri transfer komutları, bellekten veya G/Ç portlarından saklayıcılara ya da saklayıcılardan belleğe veri taşımak için kullanılırlar. Bir mikrodenetleyici tipik olarak bir veriyi bir yerden başka bir yere taşımak için diğer işlemlerden daha fazla zaman harcamaktadır. Veri transfer komutları 3 başlık altında toplanabilir. Bunlar:

- Dahili veri belleğine erişmek için kullanılanlar
- Harici veri belleğine erişmek için kullanılanlar
- Program belleğine ya da bakış tablolarına erişmek için kullanılanlar.

2.6.3.1. Dahili Veri Belleği Veri Transfer Komutları

Dahili veri transfer komutları ile 8051 mikrodenetleyicisinde dahili olarak bulunan alt RAM, üst RAM ve SFR bölgesinden veri yükleme, veri okuma işlemleri gerçekleştirilebilir. Yine bu komutlar kullanılarak SFR'de tanımlı portlardan veri giriş/çıkışı yapılmaktadır. Dahili veri belleğine erişmek için kullanılan veri transfer komutları bayrakları etkilememektedir. Kullanım şekillerine ve Byte olarak uzunluklarına göre dahili veri belleği veri transfer komutları aşağıdaki tablodaki gibidir.

Tablo 2. 5. Dahili veri belleği transfer komutları [1]

Komut		Açıklama	Bayt
MOV	A, Rn	Rn saklayıcısındaki değeri akümülatöre yükle	1
MOV	A, adres	Adresteki bilgiyi aküye yükle	2
MOV	A, @Ri	Ri'nin gösterdiği adresteki bilgiyi aküye yükle	1
MOV	A, #bilgi	Doğrudan *bilgi* verisini aküye yükle	2
MOV	Rn, A	Akümülatörü Rn saklayıcısına yükle	1
MOV	Rn, adres	Adresteki bilgiyi Rn saklayıcısına yükle	2
MOV	Rn, #bilgi	Doğrudan *bilgi* verisini Rn saklayıcısına yükle	2
MOV	adres, A	Akümülatördeki bilgiyi adrese yükle	2
MOV	adres, Rn	Rn saklayıcısının içeriğini adrese yükle	2
MOV	adres1, adres2	adres 2'deki bilgiyi adres 1'e yükle	3
MOV	adres, @Ri	Ri'nin gösterdiği adresteki bilgiyi adrese yükle	2
MOV	adres, #bilgi	Doğrudan *bilgi* verisini adrese yükle	3
MOV	@Ri, A	Akümülatörü Ri'nin gösterdiği adrese yükle	1
MOV	@Ri, adres	Adresteki bilgiyi Ri'nin gösterdiği adrese yükle	2
MOV	@Ri, #bilgi	Doğrudan *bilgi* verisini Ri'nin gösterdiği adrese yükle	2
MOV	DPTR, #bilgi16	16 bitlik *bilgi* verisini DPTR saklayıcısına yükle	3
PUSH	Adres	Adresteki bilgiyi yığına at	2
POP	Adres	Yığındaki bilgiyi adrese at	2
XCH	A, Rn	Rn ve akünün içeriklerini değiştir	1
XCH	A, adres	Adresteki bilgi ile akünün içeriğini değiştir	2
XCH	A, @Ri	Ri'nin gösterdiği adres ve akünün içeriklerini değiştir	1
XCHD	A, @Ri	Ri'nin gösterdiği adres ile akünün içeriklerinin ilk dört bitini değiştir	1

2.6.3.2. Harici Veri Belleği Veri Transfer Komutları

Harici veri belleğine erişmek için kullanılan veri transfer komutları, dolaylı adresleme yöntemini kullanır. Bu yöntemde harici bellek adresini tutmak için R0, R1 veya DPTR saklayıcılarından faydalanılır. R0 ve Rl ile 00-FFh ve DPTR ile 0000-FFFFh adres aralıklarına erişim sağlanmaktadır. Her zaman kaynak ya da hedef olarak akümülatör kullanılır. MOVX komutu harici bellek üzerinde işlem yapmak için RD, WR ve ALE uçlarını etkinleştirir. Harici bellek üzerinde çalışan tüm veri transfer komutları 2 makine çevriminde çalışır. Harici veri belleğine erişmek için kullanılan tüm veri transfer komutları aşağıdaki tabloda görülmektedir.

Tablo 2. 6. Harici veri belleği veri transfer komutları [1]

Komut	Açıklama	Bayt
MOVX A, @Ri	Ri saklayıcısının gösterdiği harici RAM adresindeki veriyi akümülatöre yükle	1
MOVX @Ri, A	Aküyü Ri'nin gösterdiği harici RAM adresine yükle	1
MOVX A, @DPTR	DPTR'nin gösterdiği harici RAM (16 bitlik adres) adresindeki bilgiyi aküye yükle	1
MOVX @DPTR, A	Aküyü DPTR'nin gösterdiği harici RAM adresine yükle	1

2.6.3.3. Program Belleği Veri Belleği Veri Transfer Komutları

MOV ve MOVX veri transfer komutları sürekli olarak işlemlerin geçici olarak kaydedildiği ve enerji kesilmesi durumunda bilgilerin kaybolduğu RAM bellek üzerinde çalışmaktaydı. Tablolar halinde önceden tanımlanmış bilgilere de erişme ihtiyacının olduğu durumlar da verilerin tekrar tekrar kullanılabilmeleri için veriler kalıcı bellek olan ROM'da tutulur. Dahili veya harici program belleğinden (ROM) bakış tablolarını okumak için aşağıdaki tabloda görülen gibi iki veri transfer komutu kullanılır.

Tablo 2. 7. Program belleği veri transfer komutları [1]

Komut	Açıklama	Bayt
MOVC A, @A+DPTR	A+DPTR'nin gösterdiği harici ROM (16 bitlik adres) adresindeki veriyi Akümülatöre yükle	1
MOVC A, @A+PC	A+PC'nin gösterdiği harici ROM (16 bitlik adres) adresindeki veriyi Akümülatöre yükle	1

2.6.4. Bit İşlem Komutlar

Bit işlemler MCS-51 ailesini diğer mikrodenetleyicilerden ayıran en güçlü özelliklerinden biridir. Dahili RAM'in belirli bir kısmı (128 adet adreslenebilir bit) ve yine SFR alanının belirli kısımları bit adreslenebilir alana sahiptir. Tüm portlar bit adreslenebilirdir ve her bir port ucu bağımsız bir uç olarak davranabilir. 8051'de kullanılan tüm bit işlem komutları aşağıdaki tablodaki gibidir.

Tablo 2. 8. Bit işlem komutları [1]

Komut		Açıklama	Bayrak	Bayt
CLR	A	Akümülatörü temizle	–	1
CPL	A	Akümülatörü tersle	–	1
CLR	C	Eldeyi sıfırla	C	1
CPL	C	Eldeyi tersle	C	1
SETB	C	Eldeyi birle (C = 1)	C	1
SETB	bit	Bit adreslenebilir RAM'deki bir bitlik veriyi birle	–	2
CLR	bit	Bit adreslenebilir RAM'deki bir bitlik veriyi birle	–	2
CPL	bit	Bit adreslenebilir RAM'deki bir bitlik veriyi tersle	–	2
MOV	C, bit	Bir bitlik adresteki veriyi elde bayrağına yükle	C	2
MOV	bit, C	Eldeyi bir bitlik adrese yükle	C	2
ANL	C, bit	Elde ile bir bitlik veriyi VE işlemine tabi tut	C	2
ORL	C, bit	Elde ile bir bitlik veriyi VEYA işlemine tabi tut	C	2

2.5.5. Program Dallanma Komutları

Program akışını kontrol etmek üzere kullanılan dallanma komutlarını, şartsız dallanma, şartlı dallanma ile alt program çağırma ve alt programdan dönme komutları olmak üzere üç grup altında toplayabiliriz. Dallanma komutları bir yere dallanmak için PC'nin değerini kalıcı olarak değiştirirken, alt program çağırma komutları ise programın diğer bir parçasının çalışmasına izin vermek için geçici olarak PC'yi değiştirir ve çağrılan program bittiğinde PC tekrar eski değerine kurulur.

2.6.5.1. Koşulsuz Dallanma Komutları

Program akışını değiştirmek için herhangi bir koşulun gerçekleşmesi gerekmez. Kullanılan koşulsuz dallanma komutunun izin verdiği adres aralığında dallanma sağlanır. Koşulsuz dallanma komutu JMP'nin, kısa (SJMP), mutlak (AJMP) ve uzun dallanma (LJMP) olmak üzere üç türü vardır. SJMP ile -127 +128 Byte aralığında bir dallanma gerçeklenirken, AJMP ile 2KByte'lık sayfa içerisinde bir dallanma ve LJMP ile de 64 KByte'lık bir alan içerisinde dallanma gerçeklenir. Hiçbir bayrağı etkilemeyen koşulsuz dallanma komutlarının kullanım şekilleri ve Byte olarak uzunlukları aşağıdaki tabloda verilmektedir.

Tablo 2. 9. Koşulsuz dallanma komutları [1]

Komut		Açıklama	Bayt
SJMP	Adres	Kısa dallanma (adrese dallan)	2
AJMP	adres11	Mutlak adresleme yönteminde kullanılır, 11 bitlik adres (2 KB) alanı içerisinde bir dallanma sağlar	2
LJMP	adres16	Uzun dallanma, 16 bitlik adres alanı içerisinde bir atlama sağlar	3
JMP	@A+DPTR	A+DPTR'nin gösterdiği adrese dallan	3
NOP		1 makine çevrim boyu işlem yapma	1

2.6.5.2. Koşullu Dallanma Komutları

8051 'de belirli bir şarta bağlı olarak program akışını değiştiren çok sayıda dallanma komutu vardır. Koşullu dallanma komutları, koşulsuz dallanma komutlarına göre daha küçük bir dallanma aralığı sunar. Bağıl adresleme yöntemi kullanıldığından program içerisinde koşullu dallanma komutunu izleyen komuttan -128 adım geri veya +127 adım ileri atlar. Tüm koşullu dallanma komutları aşağıdaki tabloda verilmiştir.

Tablo 2. 10. Koşullu dallanma komutları [1]

Komut		Açıklama	Bayrak	Bayt
JC	Adres	Eğer C = 1 ise adrese dallan	–	2
JNC	Adres	Eğer C = 0 ise adrese dallan	–	2
JB	bit, adres	Eğer bit = 1 ise adrese dallan	–	3
JNB	bit, adres	Eğer bit = 0 ise adrese dallan	–	3
JBC	bit, adres	Eğer bit = 1 ise adrese dallan sonra biti sıfırla (bit = 0)	–	3
JZ	Adres	Eğer akümülatör sıfır (A = 0) ise adrese dallan	–	2
JNZ	Adres	Eğer A = 0 değil ise adrese dallan	–	2
DJNZ	Rn, adres	Rn'i bir azalt ve Rn sıfır değilse adrese dallan	–	2
DJNZ	adres1, adres2	Adres1'deki veriyi 1 azalt, eğer sıfır değilse adres2'ye dallan	–	3
CJNE	A, adres1, adres2	Akü ve adres1'deki veriyi karşılaştır, eşit değilse adres2'ye dallan	C	3
CJNE	A, #bilgi, adres	Akü ve *bilgiyi* karşılaştır, eşit değilse adrese dallan	C	3
CJNE	Rn, #bilgi, adres	Rn ve *bilgiyi* karşılaştır, eşit değilse adrese dallan	C	3
CJNE	@Ri, #bilgi, adres	Ri'nin gösterdiği adresteki veri ile *bilgiyi* karşılaştır, eşit değilse adrese dallan	C	3

2.6.5.3. Alt Program Çağırma ve Dönüş Komutları

Alt program çağırma komutları ile PC'nin içeriği öncelikle yığına kaydedilir: Daha sonra PC yeni adres ile yüklenir. Alt program bittikten sonra yığına kaydedilen ana programda kalınan yeri gösteren adres tekrar PC'ye yüklenir. Mutlak adreslemeyi kullanan ACALL ve uzun adreslemeyi kullanan LCALL olmak üzere alt program çağırma komutlarının iki türü vardır. Eğer çağrılacak olan alt program 2 Kbyte'lık adres alanı içerisinde ise ACALL, daha büyük bir adres aralığı içerisinde ise LCALL komutu kullanılır. Alt programlar RET komutu ile, kesme ile çağrılan alt programlar ise RETI komutu ile sonlandırılır.

Tablo 2. 11. Alt program çağırma ve alt programdan dönüş komutları [1]

Komut	**Açıklama**	**Bayt**
ACALL adres11	Adres11 etiketli alt programı çağır	*2*
LCALL adres16	Adres16 etiketli alt programı çağır	*3*
RET	Alt programdan kaldığın yere dön	*1*
RETI	Kesme alt programından kaldığın yere dön	*1*

BÖLÜM 3. SİMÜLASYON KAVRAMI

Teknolojinin bu hızlı gelişimi elektronik ortamda bilgi ve haber iletimini dünyanın en ücra köşelerine dek yaymayı başarmıştır. Evler ve arabalar tam donanımlı birer ofise, tek kişilik bir atölye küresel bir işletmeye dönüşebilmiş, firmalar dünyanın değişik bölgelerinde küçük fakat etkin ofisler kullanarak, 24-saat sürekli proje üretecek etkinliğe ulaşabilmiş, uzaktan eğitim, elektronik ticaret gibi kavramlar uygulanmaya başlamıştır. Hatta, teknolojik gelişime bağlı olarak, bir ülkenin hastanesindeki bir uzmanın bir başka ülkenin hastanesindeki bir hastaya, bilgi ve haber kanalları üzerinden akıllı robotlar yardımıyla, ciddi ameliyatlar yapabilmesi düzeyine gelinmiştir.

Bu denli karmaşık sistemlerin geliştirilmesi ve kullanıma sokulması beraberinde modelleme ve simülasyon konusunu gündeme getirmiştir. Karmaşık elektronik ve haberleşme sistemlerinin tasarımı, geliştirilmesi, test edilmesi, eğitimi, vb. artık özel donanım ve yazılımlarla bilgisayar ortamında gerçeklenir olmuştur. Modelleme ve Simülasyon tasarlanacak sistem henüz elde olmadığı, gerçek test ve ölçülerin tehlikeli ve/veya pahalı olduğu gerçek test ve denemelerin yapılamadığı durumlarda vazgeçilmez bir araç haline gelmiştir.

Simülasyon, insan emeğinin geçtiği her alanda kullanıla gelmektedir. Bu nedenle simülasyonun kesin bir tarifini yapmak mümkün değildir. Fakat belli bir fikrin oluşması için aşağıdaki tanımlara bakılabilir.

3.1. Simülasyonun Tanımı

Simülasyon, gerçek bir sistemin modelini tasarlama süreci ve sistemin davranışını anlamak veya değişik stratejileri değerlendirmek amacı ile, geliştirilen bu model üzerinde denemeler yapmaktır. (Halaç, 1982)

Simülasyon, bir sistem hakkında, o sistem ile sebep-sonuç ilişkisi (girdi-işlem-çıktı ilişkisi) aynı veya benzer bir model üzerinde çalışılarak, zaman içinde gösterdiği

tepkilerin incelenmesi ve o sistem hakkında bilgi edinme aktivitesi olarak tanımlamaktadır [13].

Bir başka tanıma göre simülasyon, gerçek bir işlemin veya sistemin zamana bağlı olarak modelini tanımlayan matematiksel bir modeldir. Simülasyon ister elle, isterse bilgisayar ile yapılsın, bir sistemin yapay kayıtlarının oluşturulması ve gerçek sistemin işletim karakteristikleriyle ilgili sonuçlarının elde edilmesinde bu yapay kaydın incelenmesini kapsamaktadır. (Banks ve Carson, 1984)

Simülasyon, mevcut veya mevcut olması muhtemel sistemlerin dinamikleri üzerinde çalışma yapmak amacıyla o sistemlere ait model geliştirmek ve kullanmak olarak da tanımlanabilir [14].

Simülasyon, teorik ya da gerçek fiziksel bir sisteme ait neden-sonuç ilişkilerinin bir bilgisayar modeline yansıtılmasıyla, değişik koşullar altında gerçek sisteme ait davranışların bilgisayar modelinde izlenmesini sağlayan bir modelleme tekniğidir. Simülasyon, gerçek hayattaki olayların bilgisayar ortamına aktarılması işlemidir. Sanal ortamlar sağlayan yazılımlardır. Bir sistemin simülasyonu, bu sistemi temsil edebilecek bir model oluşturma işlemidir.

Simülasyon, gerçek bir sisteme ait model tasarlama ve sistemin davranışını anlamak veya sistemin işleyişiyle ilgili çeşitli stratejileri değerlendirme maksadı ile bu modeli kullanarak deneysel çalışma yapma işlemidir [15].

Simülasyonlar, genel tasarım formları içinde metin, test, canlandırma, seslendirme, alıştırma-uygulama gibi pek çok tasarım seçeneğinin uygulanmasına olanak tanırlar. Yaparak, yaşayarak öğrenmeyi sağlarlar.

Simülasyon, bir modelin belirli bir zaman ve mekan içerisinde işletilmesi suretiyle, model içindeki etkileşimin gösterilmesidir [16].

Eğitimsel simülasyon, bir olay veya aktivitenin etkileşim sonucu öğrenilmesini sağlayan modellemedir.

Simülasyon; önerilen veya gerçek dinamik bir sistemin modellenmesi ve zaman içindeki davranışın gözlenmesi işlemidir. Bir simülasyon çalışması, herhangi bir sistemin davranışının incelenmesi ve farklı parametrelerin çalışma durumuna etkilerinin araştırılması amacı ile yapılır. Simülasyon çalışmalarında uygulanan iki adım; model tasarımı ve deneylerdir. Model tasarımı sistemin tüm önemli durumlarını temsil eden bir modelin kurulmasıdır. Geçerli bir model kurulduktan sonra deneyler kısmı baslar. Simülasyon genellikle mevcut olmayan veya pahalı ve zor gerçekleştirilebilecek sistemlerin denenmesine imkan sağlar.

Simülasyon zaman boyutu olan bir aktivite, bir işlemdir; modelin kendisi değildir. Diğer bir ifadeyle simülasyon, bir sistem modelinin belirli bir zaman aralığında canlandırılmasıdır. Simülasyon, iş dünyasından, ekonomi ve eğitime, sosyal ve teknik bilimlerden, eğlence dünyası ve askeri uygulamalara kadar hemen her alanda kendini göstermektedir. Bunlara ek olarak, her türlü sosyal, ekonomi ve teknik bilim dallarında sayısız teknik makale, raporlar, master ve doktora tezleri simülasyon uygulamalarındaki hızlı gelişmeyi, özellikle sayısal bilgisayarların işlem gücünün artmasına paralel olarak sağlamışlardır [17].

Bir sistem hakkında bilgi edinmek veya işleyişini öğrenmek için öncelikle o sistemin bir şekilde modelini üretmek gerekir. Model, simülasyonun temelini oluşturur. Bir simülasyon uygulamasında yukarıda açıkladığımız model çeşitlerinden amaca uygun olan herhangi biri veya birkaçı birlikte kullanılmaktadır. Ancak model sadece simülasyon amacıyla kullanılmamaktadır. 'Simülasyon modeli' simülasyon çalışmasında kullanılmak maksadıyla üretilmiş model anlamına gelir. Modelin simülasyon uygulamasında göstermiş olduğu performanstan elde edilen veriler daha sonra analiz edilerek sistem davranışı hakkında model vasıtasıyla bilgi edinilmiş olur. Simülasyon, bir sistemin davranışlarını anlamak veya anlatmak için sistem özelliklerini ve davranışlarını bir model yardımıyla taklit (temsil) etmek esasına dayanan ve zaman boyutu olan dinamik bir uygulamadır. Simülasyon çevremizdeki dünyayı keşfetme ve anlamada yeni bir yol açmaktadır. Simülasyonun kullanımı ve anlaşılması kolaydır. İlgiyi arttırır ve takım çalışmasına olanak verir [17].

3.2. Simülasyonun Genel Özellikleri

Stok kontrol ve kaynak sistemleri modellemesi gibi simülasyon tekniği bakımından kesikli konum simülasyonu kapsamında incelenen modelleme çalışmalarının bilgisayarda programlanmasında dikkati çeken temel özellikler Law ve Kelton (1991) tarafından belirtilmiştir;

- 0 ile 1 arasında uniform U(0,1) dağılışından şans sayısı türetimi,
- Bilinen bir olasılık dağılışından şans değerlerinin türetimi,
- Simülasyon saatinin çalıştırılması,
- Uygun simülasyon bloklarına geçişte kontrol sisteminin kurulması,
- Simülasyon listesine kayıt ekleme, kayıt çıkarma olanakları,
- Uygun veri analiz yöntemlerinin kullanımı,
- Sonuçların yazdırılması,
- Hataların izlenmesi,

Bunlar ve kısmen ileride belirtilecek özellikler simülasyonda özel amaçlı simülasyon dillerinin kullanımını zorlamaktadır. Bu diller daha sonra simülasyon tekniklerinin kullanım alanının genişlemesine yol açmıştır. Ancak buna rağmen özel amaçlı simülasyon dilleri ile genel amaçlı programlama dillerinin arasında simülasyon senaryolarının bilgisayarda programlanması açısından uzun zamandan beri avantaj ve dezavantaj tartışmaları süregelmektedir.

3.3. Simülasyonun Kullanım Amaçları

Özel amaçlı simülasyon dilleri, düşük operasyon maliyetleri için yüksek hesaplama kabiliyetleri ve simülasyon metodolojisindeki gelişmeler, simülasyonu yöneylem araştırmasında ve sistem analizinde en çok kullanılan ve kabul edilen bir metot yapmıştır. Simülasyonun hangi şartlar altında kullanılması gerektiği birçok yazar tarafından incelenmiştir. Bunları genel olarak sınıflandırırsak, simülasyon aşağıdaki amaçlar için kullanılabilir (Bank ve Carson, 1984):

- Simülasyon, karmaşık bir sistemin içyapısını veya karmaşık bir sistemdeki alt sis-

temi incelemek için kullanılabilir,

- Bilgi, organizasyon el ve çevresel değişiklikler simüle edilebilir ve modelin davranışı üzerinde bu değişikliklerin etkileri incelenebilir,
- Bir simülasyon modelinin tasarımından elde edilen bilgiler, incelenen sistemin geliştirilmesine büyük ölçüde katkıda bulunmaktadır,
- Simülasyon girdilerini değiştirerek ve sonuçları inceleyerek, hangi değişkenlerin daha önemli olduğu ve değişkenlerin birbirlerini nasıl etkiledikleri hakkında bilgi edinilir,
- Simülasyon, analitik çözüm metodolojisini destekleyen bir bilgi verici araç olarak kullanılabilir,
- Simülasyon, uygulamadan önce yeni tasarımlar ve politikalar deneyerek durumun ne olacağını görmek için kullanılabilir,
- Simülasyon, analitik sonuçları test etmek için kullanılabilir.

3.4. Simülasyonun Avantajları ve Dezavantajları

Simülasyon çalışması problem çözmede son derece güçlü bir yardımcı olup, yaygın kullanışının çeşitli nedenleri vardır. Bunlar şu baslıklar altında derlenebilir:

- Karmaşık yapıdaki gerçek sistemleri analitik olarak inceleyerek matematiksel modellerin kurulmasındaki güçlükler.
- Simülasyon; yeni politikalar, parametreler veya çalışma koşullarının denemesine imkan sağlayarak sistem performansının bu yeni koşullar için tahmini sağlar.
- Alternatif dizaynların birbiri ile karşılaştırılmasını mümkün kılar.
- Gerçek sistemin rahatsız edilmeden, bozulmadan, tehlikeye atılmadan denenmesi sağlanır.
- İncelenen sistemin farklı zaman akışlarında ele alınması mümkündür. Örneğin, sıkıştırılmış bir zamanda çalışma hızlandırılarak sistem hakkında genel bilgi elde edinilebileceği gibi, geniş bir zaman aralığında sistem hakkında ayrıntılı bilgi edinme mümkün olabilir.

Bu avantajlara rağmen, simülasyon çalışmalarının bazı dezavantajlarının da belirlenmesi gerekir.

- Simülasyon modelleri pahalı ve geliştirilmesi zor modellerdir.
- Simülasyon modellerinin stokastik yapısı, gerçek sistemle ilgili ancak tahminlerde bulunmayı sağlar
- Simülasyon modelleri probleme en iyi çözümü bulmak yerine alternatif çözümleri karşılaştırır.
- Simülasyon sonuçlarının incelenen sistemi doğru yansıtması için modelin geçerliliği çok önemlidir.
- Simülasyonda bilgisayara olan bağımlılık, çalışmanın uzun sürmesine pahalı olmasına neden olur.

3.5. Simülasyonun Uygulama Alanları

Simülasyon, çok çeşitli alanlarda uygulama alanına sahiptir. Hillier ve Lieberman (1980), bu tekniğin geniş uygulama alanlarını belirtmek için aşağıdaki örnekleri vermişlerdir:

- İşletme politikaları ve uygulamalarındaki (bakım kapasitesi, tesislerin, yedek uçakların vb.) değişiklikleri test etmek için bir havayolu şirketi tarafından büyük bir havaalanındaki operasyonların simülasyonu,
- En iyi trafik akışını belirlemek için, trafik ışıklarının simülasyonu,
- Optimal tamir personeli sayısını belirlemek için bakım operasyonu simülasyonu,
- Bir radyasyon kalkanına yansıyan radyasyonun yoğunluğunu belirlemek için, bakım operasyonu simülasyonu,
- Bir radyasyon kalkanına yansıyan radyasyonun yoğunluğunu belirlemek için, kalkandaki yüksüz parçacıkların akış simülasyonu,
- Uygulama, kapasite ve tesislerin şekillerindeki değişiklikleri değerlendirmek için, çelik üretim operasyonunun simülasyonu,
- Ekonomik politika kararlarının etkilerini tahmin etmek için ekonomi simülasyonu
- Savunma ve saldırı silah sistemlerini değerlendirmek için büyük çaplı askeri savaşların simülasyonu,
- Büyük çaplı dağıtım ve envanter kontrol sistemlerinin tasarımını geliştirmek için bu sistemlerin simülasyonu,
- Firmanın politikaları ve operasyonlarındaki değişiklikleri değerlendirmek için

tüm firmanın genel operasyonlarının simülasyonu,

- En ekonomik düzeyde, tatmin edici servis sağlamak için, gerekli parça kapasitesini belirlemek maksadıyla bîr telefon iletişim sisteminin simülasyonu,
- En ideal baraj, elektrik santralı ve sulama işlerinin şeklini belirlemek için, ırmak havza operasyonlarının simülasyonu.

BÖLÜM 4. İŞLEVSEL SİMÜLATÖRÜN GERÇEKLEŞTİRİLMESİ

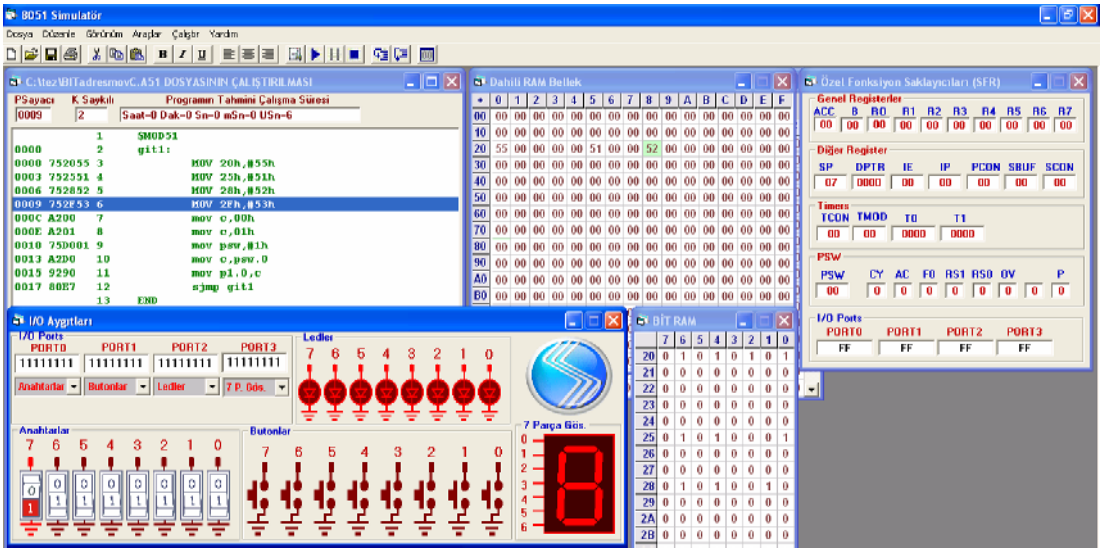
Simülatörler, kullanıcının yazdığı programların analiz edilmesini sağlayan özelleştirilmiş bir programdır. Bir simülatör, yazılımın hazır olduğu anda test edilemeyen çevre cihazlarının zamanlama karakteristiklerini modelleyebilir veya taklit edebilir. Simülatörler bellek ve kaydedicilerle birlikte, komut durma noktaları, bellek dökümleri, kaydedici kurulumları ve program izlenmesinin yanında G/Ç operasyonları ve kesme işlemlerini de taklit edebilir.

Makine kodu ile yazılmış programların orijinal işlemcisi dışında başka bir işlemci üzerinde çalıştırılabilmesi, simülatörlerin bir eğitim aracı olarak kullanılması yanında, mikroişlemcili sistemlerin geliştirilmesine paralel olarak yazılım geliştirilmesine yardımcı olmaktadır. Simülatörün bir yazılım geliştirme aracı olarak kullanılması amaçlandığında aşağıda sıralanan bazı kısıtlamalar ortaya çıkmaktadır. Bunlar;

- Simülatör, mikrodenetleyicinin çalışmasını gerçek zamanda bire bir yansıtamaz. Örnek olarak, makine dilindeki bir komutun yaptığı işi büyük makineler birden çok komutla gerçekleştirmektedir. Bu bir yazılım geliştirme kısıtlaması olmasına rağmen aslında mikrodenetleyici mimarisini ve assembly dilini öğretmek için gerçekleştirilen öğretim setleri için bir kazanımdır. Elektromekanik eğitim setlerinde çalıştırılan her komutun ne yaptığını izlemek kolay değildir, fakat simülatörde bir komutun çalıştırılması uzun zaman alacak biçimde ayarlanabilmektedir.
- Simülatörler giriş ve çıkış birimini tam olarak modelleyemezler.
- Zamanlama hataları yazılım simülasyonu ile tespit edilemezler. [11]

4.1. Simülatörün Genel Yapısı

Tasarlanan eğitim aracı ile, 8051 mikrodenetleyicilerinin yapısı, komut setinin kullanımı ve program yazımının öğretilmesi amaçlanmıştır. Simülatör Visual Basic 6.0 derleyicisiyle koda dönüştürülmüştür. (Şekil 4.1)



Şekil 4.1 8051 mikrodenetleyici simülatörünün genel görünüşü

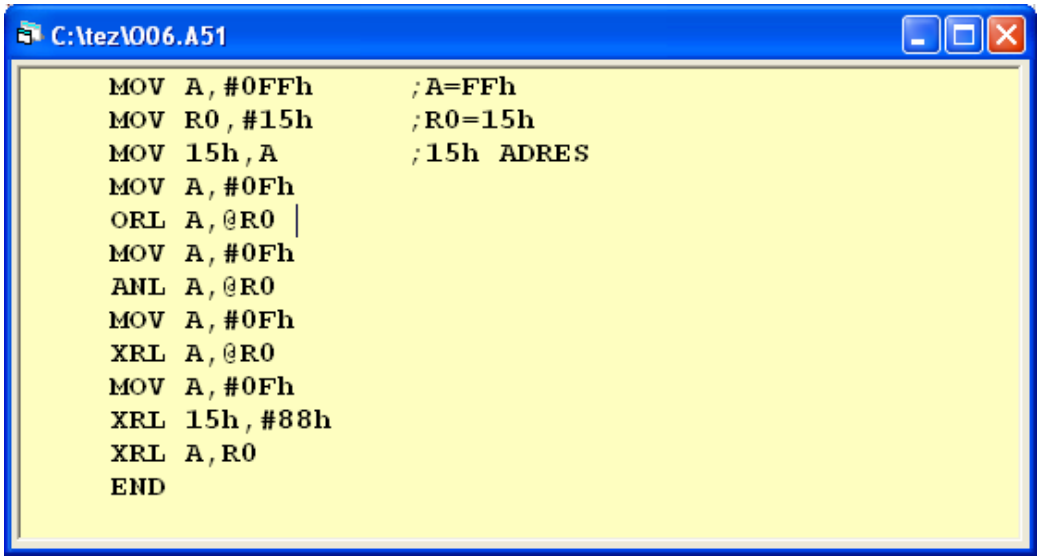
Yukarıdaki resim bir programın çalıştırılması anında simülatörden alınan bir görüntüdür. 8051 simülatör sekiz ana bölümden meydana gelmektedir. Simülatör üzerindeki işlemler menü seçenekleri kullanılarak veya bu seçeneklerin kısa yol tuş bileşimleri kullanılarak yapılmaktadır. Bu bölümler sırayla:

- Kod yazım ve düzenleme penceresi
- Kod derleme ve hata takip penceresi,
- Program çalıştırma ve takip penceresi,
- Dahili RAM bellek,
- Harici RAM bellek,
- Özel fonksiyon saklayıcıları,
- Program belleği,
- Giriş/Çıkış Aygıtları

4.1.1. Kod Yazım ve Düzenleme Penceresi

Kod yazım ve düzenleme penceresi içerisine yeni programlar yazılarak simülatör çalıştırılabileceği gibi daha önceden yazılan program dosyaları açılarak da çalıştırılabilir. Bu pencere içerisine 8051 mikrodenetleyicisinin komut seti kullanılarak assembly dilinde program yazılır. Programı yazarken assembly dilinin yazım kurallarına uyulması zorunludur. Aksi durumda, programın derlenmesi aşamasında istenmeyen hata-

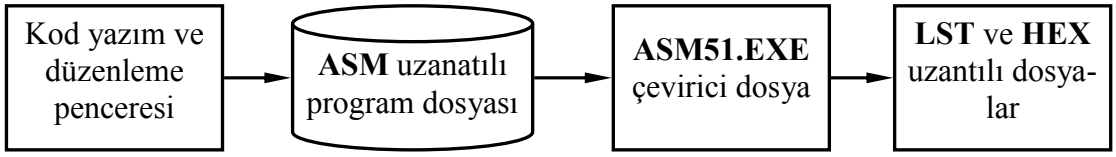
lar ortaya çıkar. Aynı anda birden fazla kod yazım ve düzenleme penceresi açık olabilir. Bu pencerelerden aktif olanı üzerinde **Derleme** yapılır. Derleme işlemi sonunda kodlarla ilgili herhangi bir hata olmadığı simülatör tarafından belirtildikten sonra kullanıcının onayı ile program çalıştırılabilir.



Şekil 4.2. Kod yazım ve düzenleme penceresi

Kod yazım ve düzenleme penceresindeki programın istenirse sisteme bağlı bir yazıcıdan çıktısı alınabilir veya bir dosya adı ile saklanabilir. Kod yazım alanında neredeyse bütün editörlerde kullanılan Bul, Değiştir, Sil, Kes, Kopyala ve Yapıştır fonksiyonları da kullanılır.

4.1.2. Kod Derleme ve Hata Takip Penceresi



Şekil 4.3. 8051 assembly komutalarının derlemesi işlemi

Kod yazım ve düzenleme penceresinde yazılan programlar bir çevirici (assembly) tarafından bir HEX uzantılı dosyası ve bir de LST dosyası oluşturur. Çevirici tarafından hazırlanan bu dosyalar herhangi bir simülatör programı tarafından veya mikrodenetleyici tarafından çalıştırılabilir. Tasarlanan simülatör OBJ uzantılı dosya içerisin-

deki assembly komutlarının mikrodenetleyici içerisindeki gerçek adres bilgilerini ve komut kodlarını kullanarak simülasyon işlemini yapmaktadır.

4.1.2.1. ASM Uzantılı Kaynak Dosyasının Hazırlanması

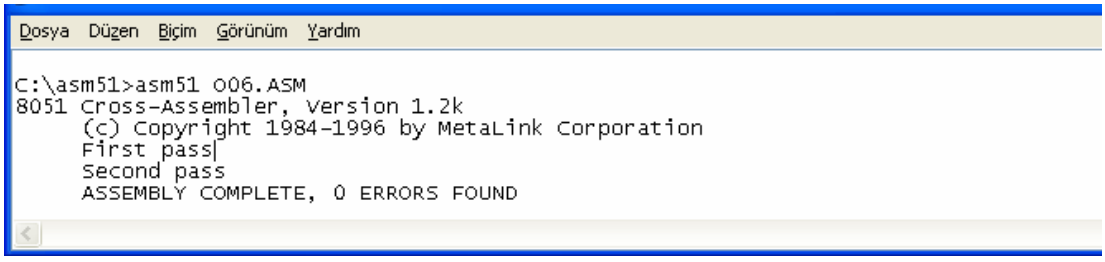
Kod yazım ve düzenleme penceresindeki programı disk üzerine saklanarak kaynak dosya oluşturulur. Bu dosya kullanılarak her zaman simülasyon işlemi tekrarlanabilir.

4.1.2.2. ASM51.EXE Assemblerının Genel Yapısı

8051 mikrodenetleyici ailesi için kullanılan ASM51, kullanım kolaylığı ve modüler programlamayı sağlayabilmesi açısından hem amatör hem de profesyonel uygulamalar için yeterlidir. Assembler ya da herhangi bir text editörde yazılan kod ASM51'de derlendikten sonra bazı işlemlerin ardından mikrodenetleyici sistemin çalıştırılabileceği HEX dosyasına dönüştürülür.

Yazılan kod bir ASM dosyasında tutulur. Kod yazma işleminin ardından derleyici ana programları derleyerek kod yazım hataları olup olmadığını kontrol ettikten sonra LIST (*.LST) dosyalarını oluşturur. Bu işlemin ardından assembler LIST dosyasından 8051 komutlarının opcode (operation code) larını oluşturur. Opcode mikrodenetleyici sistemin yazılımla, komut işlemlerinin donanım tarafından yürütülebilir hale getirilmiş halidir. Yani opcode'lar yazılım kodunun makine kodu (machine code) karşılığıdır.

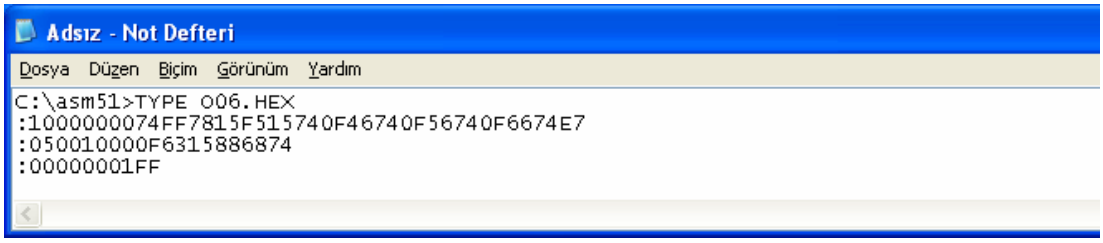
4.1.2.3. LST ve HEX Dosyasının Oluşturulması



Şekil 4.4. ASM51.EXE çevirici ile HEX ve LST dosyasının oluşturulması

Komut satırına;

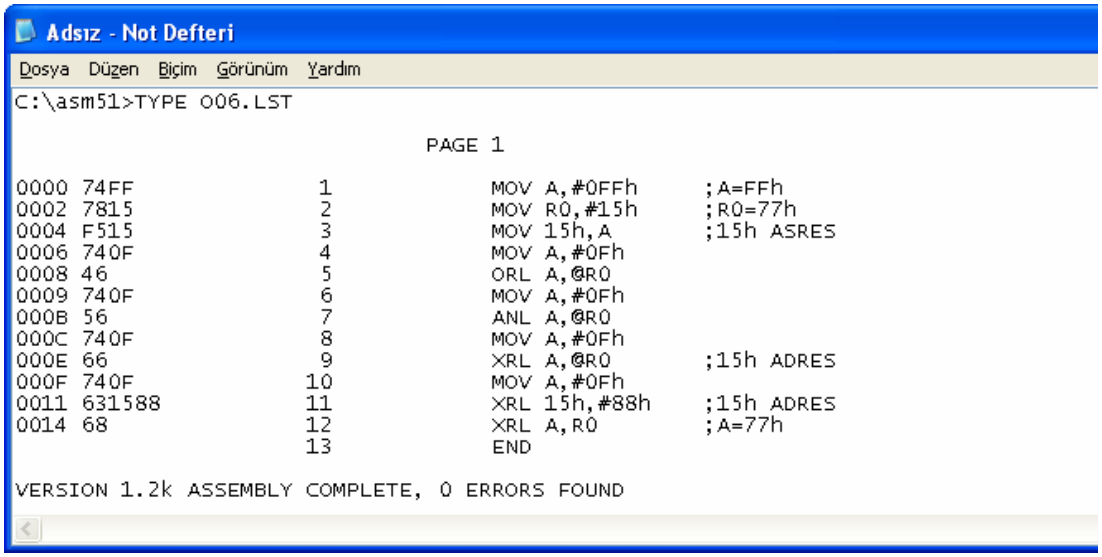
C:\asm51>asm51 O06.ASM ifadesi yazılarak derleme işlemi yapılır.



Şekil 4.5. O06.HEX dosyasının içeriği

Komut satırına;

C:\asm51>TYPE O06.HEX ifadesi yazılarak yukarıdaki gibi HEX dosyanın içeriği görüntülenir.



Şekil 4.6. O06.LST dosyasının içeriği

Komut satırına;

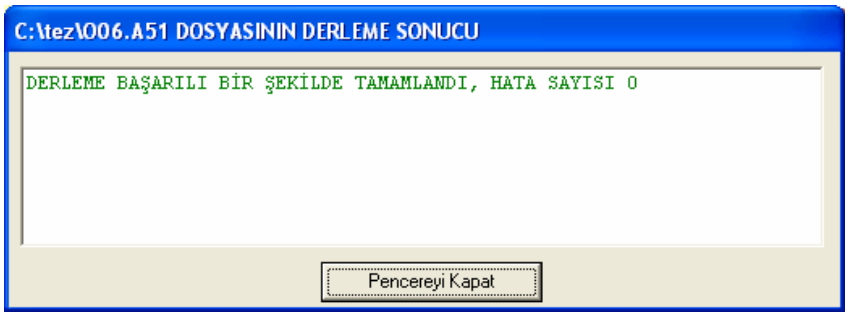
C:\asm51>TYPE O06.LST ifadesi yazılarak yukarıdaki gibi LIST dosyanın içeriği görüntülenir.

Birinci sütundaki veriler program belleğine yerleştirilen assembly (MOV, ORL, vb.) kodlarının opcode'unun adres bilgileridir. İkinci sütunda ise opcode'ların ve bu opcode'ların (assembly komutların) işleyeceği verilerin onaltılık tabandaki karşılıkla-

rı bulunur. Üçüncü sütunda program kodlarının satır numaraları bulunur. Dördüncü sütunda ise kod yazım ve düzenleme alanındaki programın aynısı bulunur.

Simülatörde birinci, ikinci ve üçüncü sütundaki veriler kullanılarak işlemler yapılmaktadır.

4.1.2.4. Hata Denetiminin Yapılması



Şekil 4. 7. Assembly programının derlenmesi ve sonucun kullanıcıya bildirilmesi

Program yazımı bittikten sonra **Çalıştır/Derle F12** komutu kullanılarak programın hata kontrolü yapılır, programda hata yoksa programın derlenmesi işlemi tamamlanır (Şekil 4.7). Bu aşamadan sonra program adım adım ve doğrudan çalıştırılabilir. Yukarıdaki pencerede hatasız yazılan bir programın sonucu görülmektedir. Derleme işleminde hatalar ortaya çıkarsa bu hataların sayısı ve hatalı satır numaraları ile birlikte hata mesajları da görüntülenir.

Hazırlanan simülatör hata mesajlarını, hatalı satır numarasını LST uzantılı dosya içerisinde alır ve kullanıcıya iletir. Yazılımda yapılan hatalar:

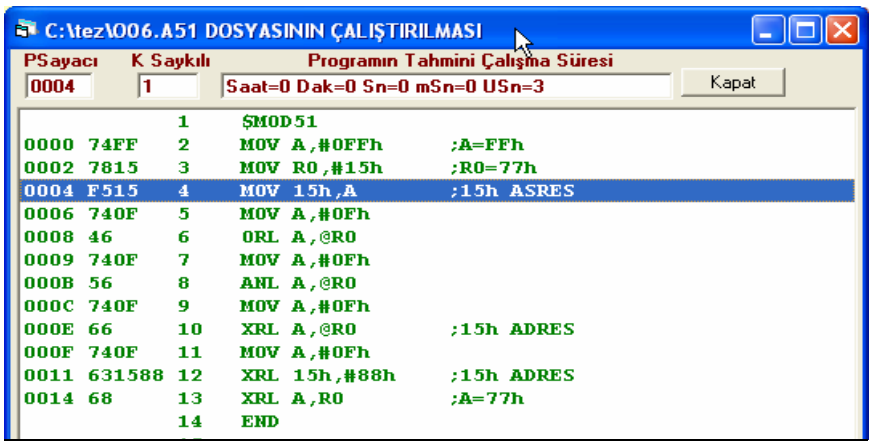
Tablo 4.1. Derleme aşamasında oluşabilecek hata mesajlarından bazıları

ERROR # 1 : Illegal character	Yazılımda kullanılmaması gereken bir karakterin kullanıldığını gösterir.
ERROR #2 : Undefined symbol	Tanımlanmamış bir sembol kullanıldığında oluşur.
ERROR #3 :Dublicate symbol	Daha önce tanımlanmış bir sembol ikinci kez tekrar tanımlanmaya çalışılırsa oluşur.

ERROR #4 :Illagal digit for radix.	Sembollerin kullanılmaması gereken bir karakter veya rakamla başlaması sonucu meydana gelir.
ERROR #5 : Number too large.	Belirtilen sayı 16 bit'i (65535) geçtiğinde oluşur.
ERROR #6 : Missing END directive	Ana program sonunda " END " kullanılmadığı zaman oluşur.
ERROR #7 : Illegal opcode / directive after label	Etiketten sonra yanlış opcode veya directive kullanıldığında oluşur.
ERROR #9 : Text beyond END directive	Program sonunda kullanılan " END " den sonra program yazılımı devam ettiğinde oluşur.
ERROR #10 : Illegal or missing expression	Programda yanlış veya eksik ifade kullanılmış demektir.
ERROR #11 : Illegal or missing expression operator	Programda yanlış ve eksik bir matematik operatörü kullanılmış demektir.
ERROR #12 :Unbalanced parantheses	Eksik parantez kullanılmış demektir.

4.1.3. Program Çalıştırma ve Takip Penceresi

Derleme işlemi başarı ile tamamlandıktan sonra program çalıştırılabilir. Hata oluşursa hataları düzeltmek üzere kod yazım ve düzeltme penceresine dönülür.

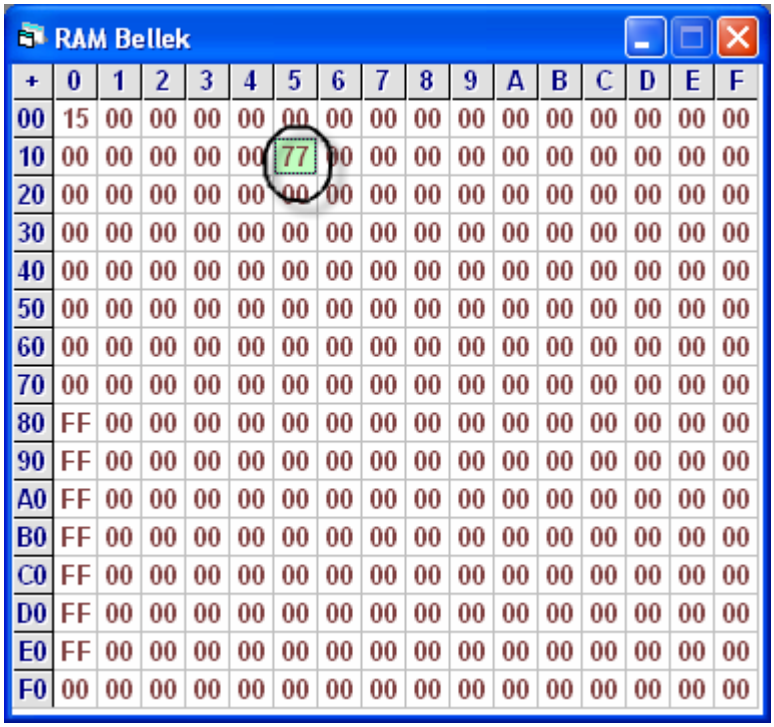


Şekil 4.8. Derlenen dosyanın çalıştırıldığı pencere

Simülatörde bu aşamaya gelindiğinde program F8 kısa yol tuşuyla adım adım veya Ctrl+F12 tuş bileşimiyle de direkt olarak çalıştırılabilir.

Bu pencere üzerinde program sayacı, komutun kaç çevrim olduğu ve program çalıştırılması aşamasında geçen süre izlenebilir.

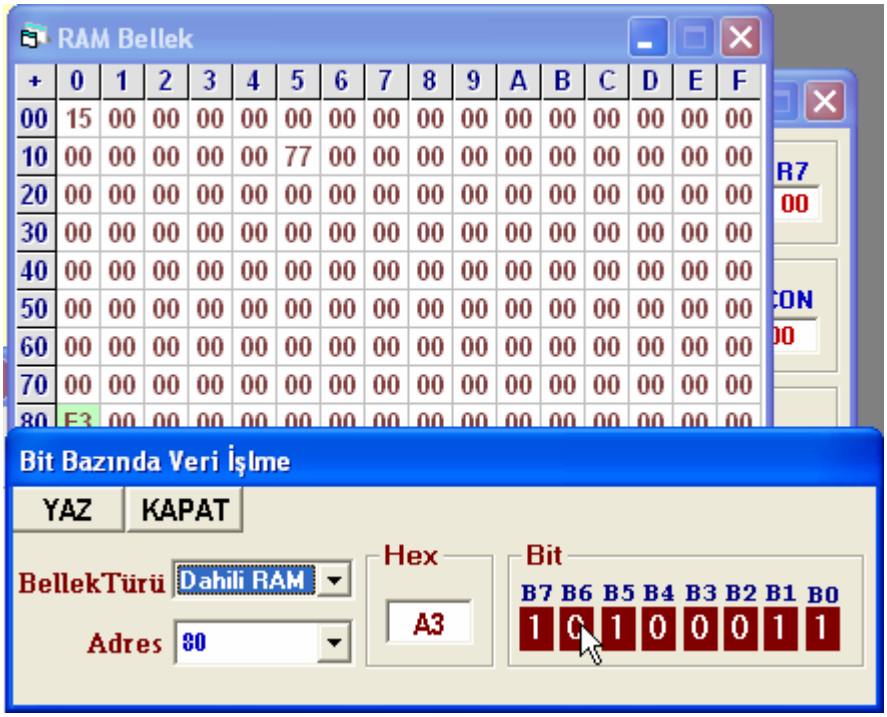
4.1.4. Dahili RAM Bellek



Şekil 4.9. Dahili RAM bellek haritası

Programın çalıştırılması sırasında RAM belleği durumunu takip etmek için kullanılır. Diğer simülatörlerden farklı olarak adım adım çalıştırma esnasında o anda çalışan komutun etkilediği bellek hücresi işaretlenir. Bu işlem komutun etkilediği bellek alanlarının takibi ve komutun etkisinin anlaşılması için etkili bir yöntem olarak tasarlanmıştır.

RAM bellek alanındaki bilgiler asembly programı çalıştırılırken istenirse kullanıcı tarafından değiştirilebilir. Bu işlem için veri değiştirilecek hücre üzerine çift tıklamak yeterlidir. Hücre üzerinde çift tıklandığında aşağıdaki şekildeki Bit Bazında Veri İşleme iletişim kutusu görüntülenir.



Şekil 4.10. Bellek hücresindeki verilerin bit bazında değerinin değiştirilmesi

Bit Bazında Veri İşleme penceresinde hücrenin adresi, hücredeki verinin HEX ve Bit bazındaki değerleri görüntülenir. Veri değerinin değiştirmek için şekildeki gibi bitlerin üzerine çift tıklanır. Verinin belleğe yazılması için **YAZ** düğmesine tıklanır.

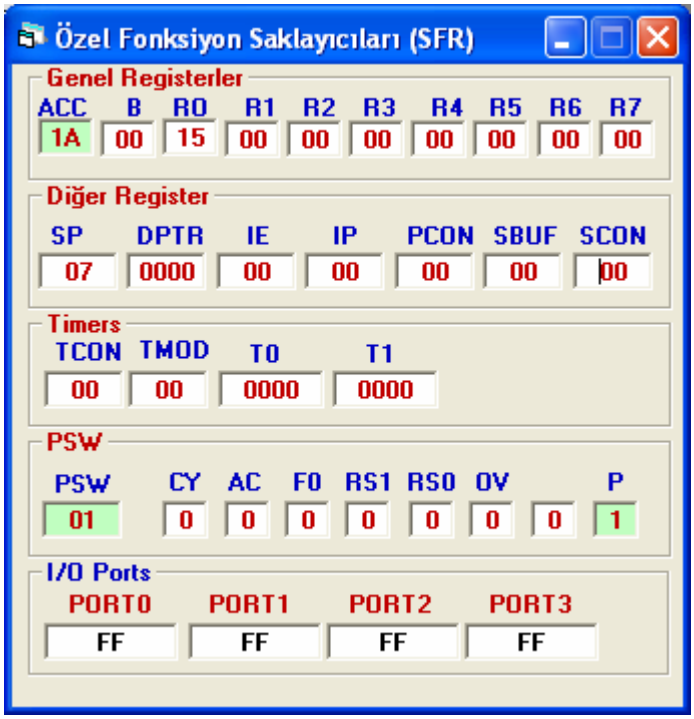
4.1.5 Harici RAM Bellek

Harici RAM bellek haritasını görüntüler. Çalışma şekli dahili RAM gibidir. Yani bellek hücresindeki değişimler takip edilebilir veya kullanıcı kendi isteğine göre bellek hücresindeki verinin değerini bit bazında değiştirebilir.

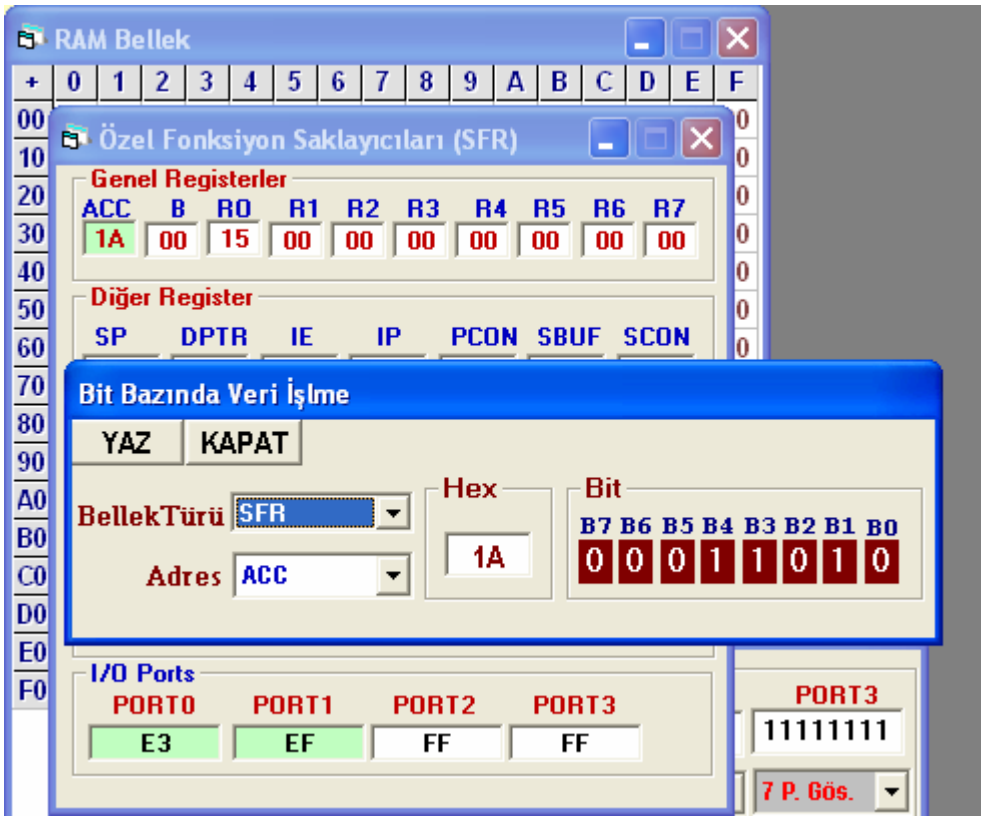
4.1.6 Özel Fonksiyon Saklayıcıları

SFR penceresi aracılığıyla kaydedicilerin içerdikleri veriler görüntülenir. Dahili RAM bellek alanında olduğu gibi buradaki bellek hücrelerinde de herhangi bir komut tarafından bir değişim meydana getirildiğinde bellek hücresi işaretlenir.

Kaydedici içerikleri aynı RAM belleklerde olduğu gibi kullanıcı tarafından bit bazında değiştirilebilir.



Şekil 4.11. Özel fonksiyon saklayıcıları



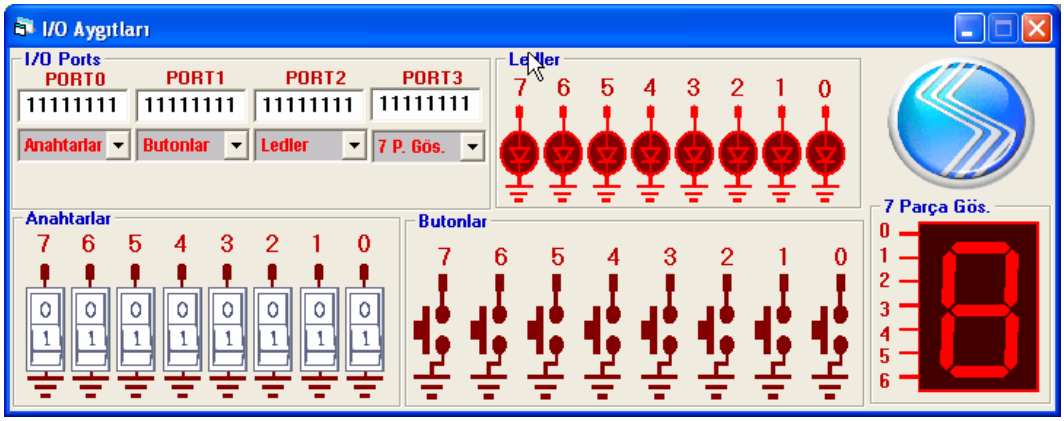
Şekil 4.12. SFR'deki verilerin değerinin bit bazında değiştirilmesi

4.1.7. Program Belleği

Program belleği penceresinde programın HEX kodları görüntülenir.

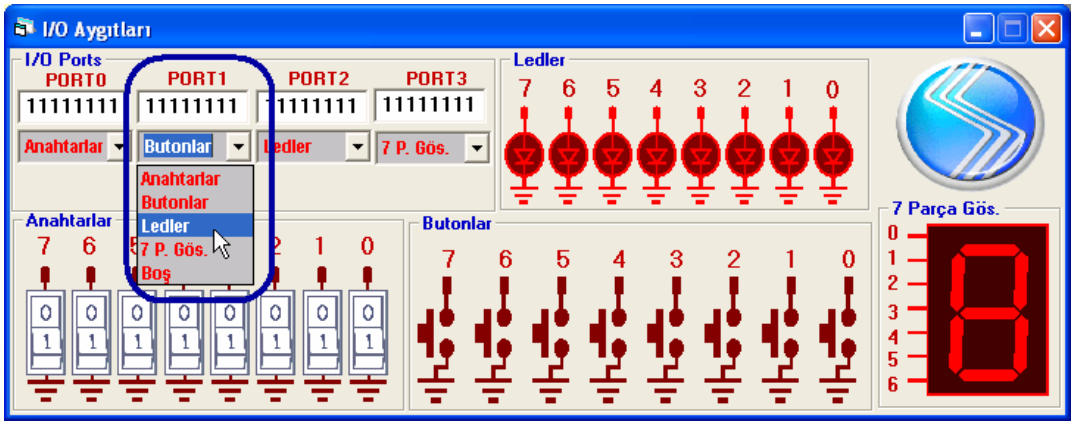
4.1.8. Giriş/Çıkış Aygıtları

Bu simülatör diğer simülatörlerden farklı kılan yapılardan birisi de aşağıdaki şekilde görüntülenen giriş/çıkış aygıtlarının fonksiyonel ve ayrıntılı bir biçimde görüntülenmesidir.



Şekil 4.13. Giriş/Çıkış portlarının genel yapısı

4.1.8.1. Giriş/Çıkış Aygıtlarının Portlara Bağlanması



Şekil 4.14. Porta bağlanana aygıtların

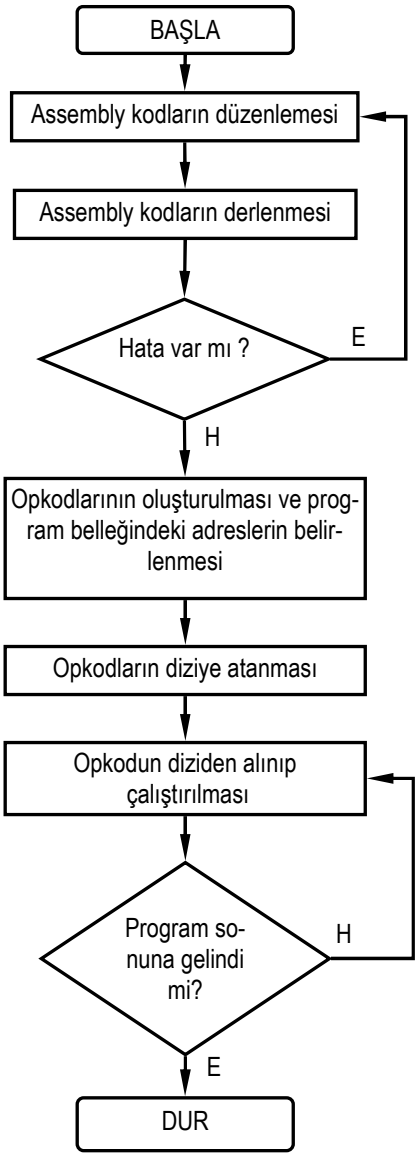
Başlangıçta Port0'a anahtarlar, Port1'e butonlar, Port2'ye ledler ve Port3'e de yedi parça gösterge bağlı olarak gelir. Kullanıcı bu aygıtları değiştirmek veya portu boşa almak isterse portların altındaki açılır liste kutusundaki aygıt isimlerinden birisini se-

çerek bu işlemi yapar. Örneğin butonlarla ledlerin yerini değiştirmek istersek önce portlardan birisini boşa almamız gerekir. Çünkü aynı anda iki porta bir aygıt bağlanamaz. Bu kurallara dikkat edilerek istenilen aygıt istenilen porta bağlanabilir.

Ledler portun durumuna göre yanar veya söner. Örneğin 11110000 bilgisine sahip bir porta ledler bağlanırsa ledlerin son dördü yanar, ilk dördü söner. Aynı etkiyi yedi parça göstergelerde de görebiliriz.

Benzer şekilde anahtarların bağlandığı porttaki veriler ise anahtarların durumuna göre değişir. Örneğin anahtarlardan ilk ikisi açık, sonrakiler kapalı ise porttaki bilgi 00000011 şeklinde değişir. Yukarıda anlatılan yapı sayesinde giriş çıkış elemanlarının bir deneme kartına bağlanan elemanlar gibi davranması sağlanmıştır. Portlara bağlı elemanlar simülasyonun herhangi bir anına çıkarılabilir (boşa alınır) veya değiştirilebilir.

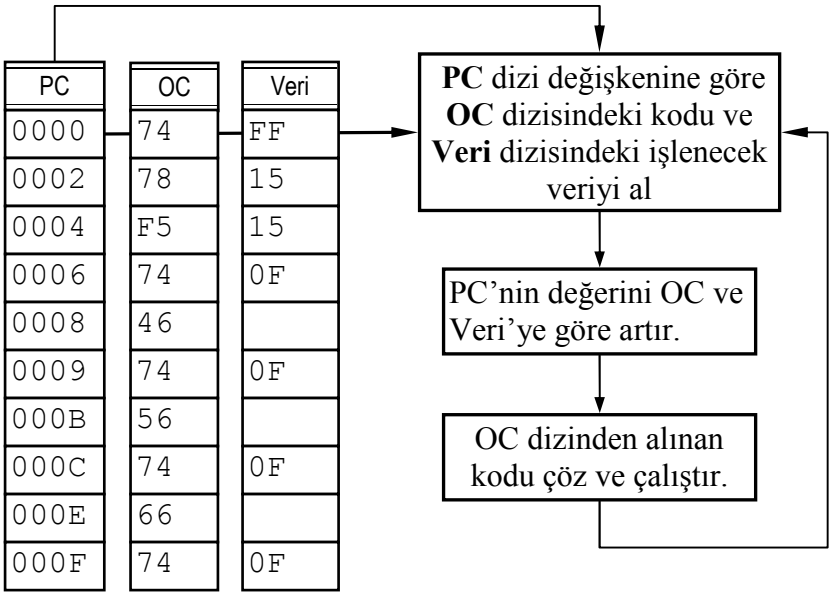
4.2. Simülatörün Çalışma Prensibinin Gösterimi



Şekil 4.15. Simülatörün çalışma yapısının akış diyagramı olarak gösterimi

Simülatörde ilk aşama assembly kodunu kullanılarak program yazmak veya daha önceden yazılan programları çağırmakla başlar. Kod yazımı tamamlandıktan sonra bir çevirici (ASM51.EXE) aracılığıyla kodların derlenmesi işlemi yapılır. Derleme sonucunda hata olup olmadığı tespit edilir. Hata varsa tekrar program koduna dönülerek hatalar düzeltilir. Program hatasız hale gelinceye kadar bu işlem tekrarlanır.

Hatalar giderildikten sonra assembly komutlarının opcode'ları, işlenecek veriler, program sayacının adresleri alınarak diziler oluşturulur. Bu işlem aşağıdaki gibi grafiksel olarak ifade edilebilir.



Şekil 4.16. Komutların çalıştırılmasının akış diyagramı ile gösterimi

Hazırlanan dizide sadece çalıştırılan satırlardaki veriler alınır. Açıklama satırları dikkate alınmaz. Çalıştırılacak program satırları için opcode'lar, veriler ve program sayacının adresleri ayrı ayrı dizilere aktarılır. Program sayacı dizi indeksi olarak kullanılır. Sistemde OC ve Veri dizileri simülasyonun temel yapı taşını oluşturur. OC dizisinden kodu alınan komutun önce kodu çözülür ve simülatörde komutla ilgi tanımlı bölüm çalıştırılır. PC ise işlenen komutun uzunluğuna göre her zaman otomatik olarak artırılır. PC otomatik olarak artırımı OC dizisinden her zaman doğru kodun alınmasını sağlar.

Yukarıdaki anlatılan işlem END komutuna gelininceye kadar devam eder. END komutu programın sonudur. END direktifinin anlaşılabilmesi için A5 kodu kullanıldı. Komutlar içerisinde A5 herhangi bir komutun kodu değildir.

BÖLÜM 5. SONUÇ VE ÖNERİLER

Hazırlanan simülatör, elektrik, elektronik ve bilgisayar alanında eğitim veren meslek yüksek okulları, teknik eğitim fakülteleri ve mühendislik fakültelerindeki mikrodenetleyici eğitim verilen derslerin laboratuar uygulaması aşamasında kullanılmak üzere tasarlanmıştır. Günümüzde artık her öğrencinin evinde bir bilgisayar hatta birçoğunun elinde bir dizüstü bilgisayar bulunmaktadır. Hazırlanan bu simülatör okullardaki klasik laboratuar ortamını öğrencilerin evlerine getirmektedir.

Okulların için büyük ölçüde ekonomik külfet getiren mikrodenetleyici laboratuarları kurmak yerine ücretsiz olarak okul veya isteyen öğrenciler verilen bu eğitim seti mikrodenetleyicili sistem tasarımı eğitimine oldukça ekonomik bir çözüm üretmektedir.

Deney setlerinin hem maliyeti yüksek, hem de kullanımının öğrenilmesi simülatöre göre daha zor olması simülatörleri cazip hale getirmektedir. Elektromekanik deney setlerinin yanlış kullanım sonucu bozulma ihtimali olması da eğitimcileri için bir başka problem teşkil ediyor. Bu laboratuarların kullanımı için bir eğitimcinin öğrencilere yardımcı olması gerekmektedir. Simülatör ise bu tür problemleri ortadan kaldırmaktadır.

Simülatörün elektromekanik deney setlerine göre avantajları:

- Mikrodenetleyici içerisinde gerçekleşen görülmeyen fakat olduğunu bilinen olayları görsel bir şekilde öğrenciye sunması.
- Kod yazımından sonra herhangi bir yardımcı çevirici (assembly) programına ihtiyaç duyulmadan kullanılabilmesi.
- Programla ilgili düzenlemelerin çok hızlı ve kolay yapılması.
- Hata bulma işlemlerinde program adımlarının takip edilmesine imkan sağladığı için hatanın hızlı bulunmasına yardımcı olur.

– Öğrencilerin eğitimlerinin laboratuar imkanları ile sınırlı kalmasını önler.
– Portlara buton, anahtar, led gibi eleman bağlantısının elle (kablo, lehim gibi) yapılmaması zaman kaybını önler. Zaten bu tür işlemler için temel elektronik bilgisinin olması gerekir. Simülatör bu ön bilgi ihtiyacını da belli ölçüde ortadan kaldırır.
– Simülatörün ücretsiz dağıtımı kurum ve öğrenciler için ayrı birer avantajdır.

Elektromekanik deney setlerine göre dezavantaj olarak gerçek zamanlı çalışmaması verilebilir. Bu simülatör mikrodenetleyicilerin fonksiyonlarının öğretimi için tasarlandığı için içerisinde bir çok animasyon bulunur. Bu da simülatöründeki komutların gerçek zamanlı çalışmasını engeller.

Bundan sonraki çalışmalarda hem gerçek zamanlı çalışabilen hem de fonksiyonel olan bir simülatör için yeni bir çalışma yapılabilir. Giriş/çıkış aygıtlarının sayısı artırılabilir.

KAYNAKLAR

[1] Özcerit, A. T, Çakıroğlu, M, Bayılmış, C, 8051 Mikrodenetleyici Uygulamaları, Papatya Yayıncılık Eğitim, İstanbul, 2005

[2] Smith, M. R., Cheng, M., 1996, Use of "Virtual" (simulated) hardware devices in microprocessor laboratories and tutorials, Frontiers in Education Conference, FIE'96, 26th Annual Conference, Proceedings of., Vol. 3, Page(s): 1181-1185.

[3] Caldwell, C. W., Andrews, D. L., and Scott, S. S., 1995, A Graphical Microcomputer Simulator for Classroom Use, Frontiers in Education Conference, 1995. Proceedings, Vol. 2, Page(s): 3b3.9-3b312.

[4] Lovergrove, W. P., 1996, A Microprocessor Trainer Simulator, Proc. Of the 261*1 Frontiers in Education Annual Conference, Vol. 2, Page(s): 506-509.

[5] Giurgiutiu, V., Lyons J, Rocheleau D, Liu, W., 2005, Mechatronics/ microcontroller education for mechanical engineering students at the University of South Carolina, Columbia, Mechatronics 15 (2005) 1025–1036.

[6] Reshadi, M.,Mishra, P., Dutt, N., 2003, Instruction Set Compiled Simulation: A Technique For Fast And Flexible Instruction Set Simulation, Anaheim, California, USA.

[7] Türkeli, B., Intel Mc8051 Mikrodenetleyicisinin PC'de Simülasyon ile Eğitimi, İstanbul, 1995

[8] Süslü, İ. H. , 8096 Mikrodenetleyicisinin PC'de simülasyon ile Eğitimi, İstanbul, 1995

[9] Nartkaya, M., Intel 8085 Mikroişlemcisinin Simülasyonu, Ankara, 1996

[10] Küçük, S., PIC16C65 Serisi Mikrokontrolörün PC'de simülasyon ile Eğitimi, İstanbul, 1998

[11] TOPALOĞLU, N, PC Tabanlı Fonksiyonel Mikroişlemci Simülatörü Tasarımı ve Gerçekleştirilmesi, Ankara, 2002

[12] www.ieee.itu.edu.tr

[13] Fishvvick, P.A., Computer Simulation: The Art and Science of Digital World Construction, http://www.cise.ufl.edu/~fishwick/introsim/paper.html, Computer & Information Science and Engineering Department, University of Florida, CSE 301, Gainesville, FL 32611, USA, 1995

14] SHANNON, RE., Systems Simulation (The Art and Science), Prentice Hall Inc., New York,1975.

[15] HOOVER, S.V., Perry, Simulation A Problem Solving Approach, Addison Wesley,1980.

[16] FİLDİŞ, Y., 2000, Muharebe Modelleme ve Simülasyon Sistemleri, MEBS İletişim Dergisi, 10, s 32, Ankara, 2000.

[17] Demir, M., Eğitimde Simülasyon/Simülatör Uygulamaları: Hareketli Bir Helikopter Uçuş Eğitim Simülatörünün İncelenmesi ve Sınıflandırılması; Örnek Bir Bilgisayar Simülasyonu Uygulaması, 2001.

[18] Gümüşkaya, H, Mikroişlemciler ve 8051 Ailesi, Alfa Yayınları, İstanbul, 2004

ÖZGEÇMİŞ

Necat GÜNEY 1974 yılında Ordu'da doğdu. İlk ve orta öğrenimini Ordu Merkez İlköğretim okulunda tamamladı. Ordu Endüstri Meslek Lisesi ve Teknik Lisesinin Teknik Lise Elektronik bölümünü 1992 yılında bitirdi. 1992-1993 eğitim öğretim yılında Gazi Üniversitesi Teknik Eğitim Fakültesinin Elektronik Bilgisayar Eğitimi bölümünde üniversite eğitimine başladı. 1995-1996 eğitim öğretim yılında üniversite eğitimini tamamladı. 1996 yılında Bolu Merkez İzzet Baysal Anadolu Teknik Lisesi, Teknik Lise ve Endüstri Meslek Lisesi'nde öğretmenliğe başladı. Halen aynı okulda Bilgisayar Bölüm şefi olarak görev yapmaktadır.