

**T.C.  
SAKARYA ÜNİVERSİTESİ  
FEN BİLİMLERİ ENSTİTÜSÜ**

**MİKROİŞLEMCİLER ve MİKRODENETLEYİCİLER  
DERSİNİN BİLGİSAYAR DESTEKLİ EĞİTİME  
UYARLANMASI**

**YÜKSEK LİSANS TEZİ**

**Aytaç KAYA**

**Enstitü Anabilim Dalı : ELEK.BILG.EĞT.  
Tez Danışmanı : Prof. Dr. Hüseyin EKİZ**

**Temmuz 2006**

**T.C.  
SAKARYA ÜNİVERSİTESİ  
FEN BİLİMLERİ ENSTİTÜSÜ**

**MİKROİŞLEMCİLER ve MİKRODENETLEYİCİLER  
DERSİNİN BİLGİSAYAR DESTEKLİ EĞİTİME  
UYARLANMASI**

**YÜKSEK LİSANS TEZİ**

**Aytaç KAYA**

**Enstitü Anabilim Dalı : ELEK.BILG.EĞT.**

**Bu tez 24 / 07 / 2006 tarihinde aşağıdaki jüri tarafından Oybirliği ile kabul edilmiştir.**

-----  
Jüri Başkanı

-----  
Jüri Üyesi

-----  
Jüri Üyesi

## TEŐEKKÜR

Çalıřmalarımda gösterdiđi destek ve anlayıřtan dolayı tez danıřmanım Sn. Prof. Dr. Hüseyin EKİZ'e, yüksek lisansımı tamamlamam konusunda yardımlarını esirgemeyen Sn. Arař. Gör. Hasan KAÇAMAK'a, tez projesinin řekillenmesinde deđerli fikir ve bilgilerini katan Sn. Yrd. Doç. Gürsel DÜZENLİ ve Sn. Arař. Gör. Ahmet KARACA'ya, tezimin okunmasında gösterdiđi titizlik ve katkılarından dolayı Sn. Yrd. Doç. Dr. Mustafa ALTUN'a çok teőekkür ederim.

Temmuz 2006

Aytaç KAYA

## İÇİNDEKİLER

TEŞEKKÜR.....	ii
İÇİNDEKİLER.....	iii
ŞEKİLLER LİSTESİ .....	vi
TABLolar LİSTESİ.....	viii
ÖZET .....	ix
SUMMARY .....	x
BÖLÜM 1.	
GİRİŞ .....	1
BÖLÜM 2.	
INTEL 8085 MİKROİŞLEMCİLER .....	9
2.1. Mikroişlemci Kavramı .....	9
2.2. Mikroişlemcilerin Tarihçesi .....	11
2.3. Intel 8085 Mikroişlemcisinin Özellikleri .....	13
2.4. Intel 8085 Mikroişlemcisi İç Yapısı.....	14
2.4.1. Aritmetik - mantık birimi .....	15
2.4.2. Kaydedici dizisi .....	17
2.4.3. Zamanlama ve kontrol birimi .....	18
2.4.4. Komut kaydedici ve komut kod çözücüsü .....	18
2.4.5. Kesme ve seri giriş / çıkış kontrolü devreleri.....	19
2.5. Intel 8085 Mikroişlemcilerinde Bulunan Kesmeler.....	19
2.6. Intel 8085 Mikroişlemcilerinde Adresleme Yöntemleri .....	23
2.7. Programlanabilir Çevre Birimi 8255.....	25
2.8. 8085 Mikroişlemci Komut Seti .....	30
BÖLÜM 3.	
PIC16F84 MİKRODENETLEYİCİLER .....	33
3.1. Mikrodenetleyici Kavramı.....	33

3.1.1.	Mikrodenetleyicilerin genel özellikleri .....	33
3.1.2.	Neden PIC mikrodenetleyiciler .....	35
3.1.3.	Neden PIC16F84 .....	36
3.2.	Bellek Organizasyonu .....	37
3.2.1.	Bellek türleri .....	37
3.2.2.	Program belleğinin organizasyonu .....	38
3.2.3.	Veri belleği organizasyonu.....	38
3.2.4.	Özel fonksiyon kaydedicileri.....	39
3.2.5.	Yığın .....	41
3.3.	Giriş / Çıkış Portları .....	42
3.4.	TIMER0 Modülü ve TMR0 Kaydedicisi .....	44
3.4.1.	Timer0 modülü .....	44
3.4.2.	TMR0 kesmesi.....	46
3.5.	DATA EEPROM .....	47
3.6.	PIC16F84 Komut Seti .....	48
3.6.1.	Adresleme yöntemleri.....	50
<b>BÖLÜM 4.</b>		
SİMÜLASYON .....		52
4.1.	Sanal Laboratuvarlar .....	52
4.2.	Macromedia Flash.....	55
4.3.	Mikroişlemci – Mikrodenetleyici Simulasyonları .....	58
4.3.1.	Assembler tasarımı .....	59
4.3.2.	Hata ayıklama ve test etme.....	62
<b>BÖLÜM 5.</b>		
8085 SİMÜLATÖRÜ TASARIMI ve UYGULAMASI .....		63
5.1.	Genel Tasarım.....	63
5.2.	Editör Tasarımı .....	64
5.2.1.	Derleyici tasarımı.....	65
5.2.2.	Derleme hata mesajları.....	67
5.3.	Hata Ayıklama ve Test Etme.....	69
5.4.	Animatör.....	73
5.4.1.	İş kodu okuma .....	76
5.4.2.	Bellek okuma ve bellek yazma makine saykalları .....	78

5.5. Deney Devreleri ve Kullanımı.....	80
5.5.1. Trafik ışık kontrolü uygulamaları.....	82
5.5.2. 8-bitlik led paneli deney modülü.....	84
5.5.3. Yedi parçalı gösterge uygulaması.....	85
5.5.4. Adım motoru deney modülü.....	86
5.5.5. Anahtarların giriş elemanları olarak kullanılması .....	87
<b>BÖLÜM 6.</b>	
PIC16F84 SİMÜLATÖRÜ .....	88
6.1. Genel Tasarım.....	88
6.2. Editör Tasarımı .....	89
6.2.1. Derleyici tasarımı.....	90
6.2.2. Derleme hata mesajları.....	92
6.3. Hata Ayıklama ve Test Etme.....	94
6.4. Animatör.....	99
6.4.1. Komut saykılı ve boru akışı.....	100
6.5. Deney Devreleri ve Kullanımı.....	103
6.5.1. Trafik ışık kontrolü uygulamaları.....	106
6.5.2. 8-bitlik ledler .....	106
6.5.3. Yedi parçalı gösterge uygulaması.....	107
6.5.4. Adım motoru .....	107
6.5.5. Anahtarların giriş elemanları olarak kullanılması .....	107
<b>BÖLÜM 7.</b>	
SONUÇ ve ÖNERİLER.....	109
KAYNAKLAR.....	111
EKLER.....	114
EK-A 8085 MİKROİŞLEMCİSİ KOMUTLARI.....	114
EK-B PIC16F84 ÖZEL KAYDEDİCİLERİ ve AÇIKLAMALARI.....	118
EK-C PIC16F84 KOMUTLARI ve AÇIKLAMALARI .....	122
ÖZGEÇMİŞ.....	135

## ŞEKİLLER LİSTESİ

Şekil 2.1 Bir mikro işlemci sisteminin temel bileşenlerinin blok diyagramı .....	10
Şekil 2.2 Intel 8085 mikroişlemcisinde bulunan pinler .....	13
Şekil 2.3 8085 mikroişlemcisi işlevsel blok şeması.....	15
Şekil 2.4 8085A Durum kaydedici formatı .....	16
Şekil 2.5 Intel 8085 mikroişlemcisinde bulunan kaydediciler .....	17
Şekil 2.6 SIM Komutu bit formatı .....	22
Şekil 2.7 Kesme yetkilendirme ve maske kaldırma .....	22
Şekil 2.8 RIM komut bit formatı.....	23
Şekil 2.9 8255 PIA Yongası.....	26
Şekil 2.10 8255 PIA iç yapısı .....	27
Şekil 2.11 Mod tanımları ve yol arayüzü .....	28
Şekil 2.12 Mod1 girişleri/çıkışları .....	29
Şekil 2.13 Mod 2.....	29
Şekil 3.1 Mikrodenetleyici genel blok diyagramı .....	34
Şekil 3.2 PIC16F8X Blok Diyagramı .....	35
Şekil 3.3 Program Bellek Haritası ve Yığın (Microchip 2001) .....	37
Şekil 3.4 Kaydedici Dosya Haritası (Microchip 2001).....	39
Şekil 3.5 Farklı durumlarda PC'nin yüklenmesi.....	40
Şekil 3.6 Direkt ve Dolaylı Adresleme .....	41
Şekil 3.7 Yığının dairesel tampon gösterimi.....	42
Şekil 3.8 TRIS kaydedicilerinin yapısı ve bitlerin işlevleri .....	44
Şekil 3.9 TMR0 / WDT Prescaler Blok Diyagramı .....	45
Şekil 3.10 TMR0 Zamanlaması: Dahili Saat / Prescalersiz .....	46
Şekil 3.11 TMR0 Zamanlaması: Dahili Saat / Prescaler 1:2 .....	46
Şekil 3.12 TMR0 Kesme Zamanlaması .....	47
Şekil 4.1 FlashMX 2004 Text kutusu için properties paneli.....	57
Şekil 4.2 FlashMX 2004 genel pencereler görünümü.....	58
Şekil 4.3 Assemblerin birinci geçiş algoritması.....	61
Şekil 4.4 Assemblerin ikinci geçiş algoritması.....	61

Şekil 5.1 8085 simülatörü araç çubuğu .....	63
Şekil 5.2 Derleme işlemi sonuç listesi ve durma noktaları seçimi.....	66
Şekil 5.3 Komut editörü.....	68
Şekil 5.4 Program çalıştırma seçenekleri .....	70
Şekil 5.5 İşlem kütüğü penceresi .....	71
Şekil 5.6 Program, veri ve yığın bellek içeriklerini gösteren bellek penceresi .....	72
Şekil 5.7 Mikroişlemci mimarisi ve bellek elemanlarını içerek animasyon penceresi .....	74
Şekil 5.8 İşkodu alma işleminde oluşan olaylar.....	77
Şekil 5.9 Bellekten mikroişlemciye bilgi aktarımı işleminin zaman diyagramı. ....	78
Şekil 5.10 Mikroişlemci deney uygulamaları penceresi .....	80
Şekil 5.11 Uygulama modülü seçim penceresi .....	82
Şekil 5.12 Yaya geçidi .....	83
Şekil 5.13 Tek yönlü yaya kavşağı .....	83
Şekil 5.14 Sola dönüşün az olduğu kavşak .....	83
Şekil 5.15 Tek yönlü T kavşak.....	84
Şekil 5.16 Üç adet 8-Bitlik LED paneli .....	85
Şekil 5.17 Yedi parçalı gösterge uygulaması.....	85
Şekil 5.18 Adım motoru ve iç yapısı.....	86
Şekil 5.19 Anahtar grubu .....	87
Şekil 6.1 PIC16F84 simülatörü araç çubuğu .....	88
Şekil 6.2 Derleme işlemi sonuç listesi ve durma noktaları seçimi.....	91
Şekil 6.3 Editörde komutların ayrıştırılması .....	93
Şekil 6.4 Program çalıştırma seçenekleri .....	95
Şekil 6.5 İşlem kütüğü penceresi özel dosya kaydedicileri kısmı.....	96
Şekil 6.6 İşlem kütüğü penceresi dosya kaydedicileri kısmı .....	97
Şekil 6.7 Program belleğinde derlenmiş bir programın gösterilişi .....	98
Şekil 6.8 Tüm bellek gözlerini bir anda görebildiğimiz EEPROM bellek penceresi .....	98
Şekil 6.9 Tüm bellek gözlerini bir anda görebildiğimiz RAM bellek penceresi.....	99
Şekil 6.10 PIC16F84 mikrodnetleyici animasyon penceresi.....	100
Şekil 6.11 Mikrodnetleyici komut saykılı ve komut boru akışı .....	101
Şekil 6.12 PIC16F84 mikrodnetleyici deney uygulamaları penceresi .....	104
Şekil 6.13 Uygulama modülü seçim penceresi .....	105
Şekil 6.14 8-Bitlik LED paneli .....	106
Şekil 6.15 Yedi parçalı gösterge uygulaması.....	107
Şekil 6.16 Anahtar grubu .....	108



## TABLolar LİSTESİ

Tablo 2.1 En çok kullanılan 8-bitlik mikroişlemciler .....	12
Tablo 2.2 8085A mikroişlemcisinde bulunan pinlerin işlevleri.....	14
Tablo 2.3 I8255 içindeki alt birimlerin seçim yöntemi.....	26
Tablo 2.4 PIA'da denetim kütüğünün modları belirlemesi.....	28
Tablo 2.5 Birbirinden farklı komut uzunluklarına sahip üç örnek komut.....	30
Tablo 2.6 8085'de bulunan kaydedicilerin kodlanması .....	31
Tablo 3.1 PORTA – PORTB Fonksiyonları .....	43
Tablo 5.1 Adım motoruna verilecek kutuplaşma gerilimleri.....	86

## ÖZET

Anahtar Kelimeler: Bilgisayar Destekli Öğretim, Uzaktan Öğretim, 8085 Mikroişlemci, PIC16F84 Mikrodenetleyici, Mikroişlemci Simülasyonu

İçinde bulunduğumuz bilişim çağında Elektronik, Bilgisayar, Mekatronik, vb. alanlar öne çıkmaktadır. Elektronik / Bilgisayar / Mekatronik ile ilgili alanlarda eğitim gören çeşitli seviyelerdeki öğrenciler yanında, teknolojik ürün geliştiren tasarımcılar veya üreticiler için en önemli konulardan biri Mikroişlemci ve Mikrodenetleyici eğitimidir.

Öğrenim, tasarım, üretim ve imalat aşamalarında bilgisayar kullanımının yaygınlaşması, mikroişlemci/mikrodenetleyici öğretiminde bilgisayar kullanımı kavramını ortaya çıkarmış bulunmaktadır. Mikroişlemcilerde/mikrodenetleyiciler’de kullanılan Assembly programlama dilinin simülasyon ve animasyonlar kullanılarak bilgisayarlar yardımıyla öğretimi yaygın bir yöntem olarak görülmektedir.

Uzaktan eğitimin yaygınlaşması ve başarılı olması için öğretimin vazgeçilmez unsuru olan laboratuarlara her zaman ve her yerden erişilebilmesi önemli bir aşama olacaktır.

Bu çalışmada, yaygın kullanıma sahip Macromedia Flash programı kullanılarak 8085 mikroişlemcisi ve PIC16F84 mikrodenetleyicisi için komut editörü ve yorumlayıcısı yazılmıştır. Yazılan editörde yorumlanan komutların mikroişlemcinin/mikrodenetleyicinin iç mimarilerinde ve modüler yapıdaki çevre birimleriyle (7 parçalı gösterge, step motor, trafik lambaları, led v.b.) çalışmasının gerektiğinde adım-adım simüle edilmesini sağlamaktadır. Bu uygulama kullanıcıların mikroişlemci/mikrodenetleyicilerde komut yazılımı ve yorumlanması ile ilgili öğrenimlerini görselleştirerek kolaylaştırılması amaçlanmaktadır.

Macromedia Flash programının sağladığı imkânlardan yararlanarak, dersleri kullanıcılara etkileşimli ve hareketli hale getirerek sunmak, böylece hem dersleri sıkıcılıktan kurtarmak, hem de temel işlemleri kullanıcıların kendilerinin yapıp görmesi için derslere canlandırma bileşeni katılması hedeflenmektedir.

# **TRANSFERRING MICROPROCESSOR AND MICROCONTROLLER LESSONS TO COMPUTER AIDED EDUCATION**

## **SUMMARY**

Keywords: Computer Aided Learning, Distance Learning, 8085 Microprocessor, PIC16F84 Microcontroller, Simulation of Microprocessor

Electronics, computer, mechatronics etc. areas are the leading ones in this information age. Microprocessor and microcontroller education is one of the most important subjects for the students of Electronic, Computer and Mechatronic Departments and also for the designers and manufacturers who develop technological products.

Widespread usage of computer at the phases of teaching, design, production and manufacturing make it also a way at the microprocessor and microcontroller education. Education of assembly language used in the microprocessors and microcontrollers with simulations and animations is a general education method.

For becoming widespread and for the success of distance learning, it would be the important stage to reach the laboratories which is the necessary element of education always and all ways.

In this study, it is coded command editor and assembler for 8085 microprocessor and PIC16F84 microcontroller with widely used Macromedia Flash program. Coded editor simulates running assembled commands inside microprocessor / microcontroller with the modular structured peripherals (7 pieces indicator, step motor, traffic lambs, led etc.) whether step by step. This application makes coding and assembling microprocessors/microcontrollers easy for the users' learning by visualizing.

Using Macromedia Flash possibilities, it is aimed presenting the lesson interactively and active to the students while having the lesson out of boring one and making them doing the main operations by themselves.

## BÖLÜM 1. GİRİŞ

Teknolojinin sürekli gelişmesiyle birlikte eğitim ortamlarında yeni teknolojilerin kullanılması ve ortaya çıkan yeni teknolojilerin kaliteyi artırma açısından ihtiyaç halini alması, “Öğretim Teknolojisi” kavramını ortaya çıkartmış ve bu kavramın önemini arttırmıştır. Öğretim teknolojisi; belirlenmiş hedefler uyarınca, daha etkili bir öğretim için gerekli tüm bilişim teknolojilerinin birlikte kullanımı, öğrenme-öğretme sürecinin bu bağlamda tasarlanması, uygulanması ve değerlendirilmesi olarak tanımlanabilir. Bilişim teknolojileri açısından bakıldığında, teknik özellikler yanında, öğretim ortamlarında etkileşimli bir şekilde kullanılan ve her geçen gün gelişmeye devam eden eğitsel yazılımlar, eğitimde hızla ön plana çıkmaktadır [1].

Günümüzde teknolojik gelişmelerin çok hızlı olduğu ve sürdürüldüğü bir zaman dilimi yaşanmaktadır. Özellikle bilgisayar ve ağ sistemlerinde yaşanan gelişmelere paralel olarak internet teknolojisinde önemli yenilikler ve ilerlemeler sağlanmaktadır. İnternet teknolojisinin uzaktan eğitimde kullanılması sonucu, eğitime farklı bir boyut getirilmiştir [2]. İnternet ve Web kullanılarak yapılan eğitim, sağladığı olanaklar açısından öğrenme ve çalışma yöntemleri üzerinde devrim sayılabilecek yenilikler getirmektedir [3].

İnternetin günümüzde bilginin yayılmasında en önemli araçlardan birisi olduğu açıktır. Bilginin sunum çeşitliliği, sunum hızı, sunum kapasitesi ve benzeri olanaklar açısından İnternet’in diğer araçlara oranla daha üstün olduğu bilinen bir gerçektir [4]. Bilim ve teknolojiden tam olarak faydalanarak, insanımızın doğru içerik, yöntem ve tekniklerle çok yoğun bir şekilde eğitime tabi tutulması gerekmektedir. Günümüzde bu amaca yönelik olarak kullanılan yeni bilgi teknolojileri arasında televizyon, video disk, video text, etkileşimli video, telekonferans, uydular, bilgisayar, bilgisayar ağları, bilgisayar ağlarının çoklu bağlantısı olan İnternet ve Web ortamları yer

almaktadır [3].

Dünyada, mühendislik ve teknik eğitimde web tabanlı olarak verilen dersler sürekli yaygınlaşmakta ve önemi artmaktadır. İnternetin Türkiye’de yaygınlaşmasıyla birlikte web tabanlı eğitim çalışmaları ülkemizde de ivme kazanmıştır [2]. Bu ivme dünyada olduğu gibi ülkemizde de birçok üniversite ve şirketi bu konuda çalışma yapmaya sevk etmiştir [3].

Son yıllarda İnternet teknolojisinde yaşanan gelişmeler, burada kullanılan multimedya olanaklarının mesleki-teknik öğretimdeki derslerin öğretilmesi amacı ile kullanılmasına imkan tanımaktadır. Yazı tabanlı veya etkileşimli şekilde eğitim amacıyla kullanılan çalışma materyallerinin öğrencilere sunulması ve karşılığında alınan geri besleme Web’in önemini daha da arttırmaktadır. Bu amaca yönelik olarak, Web üzerinde farklı tasarım uygulamaları etkileşimli olarak tek bir arabirim altında toplanabilmekte ve bu özellik hızlı değişen bilginin büyük kitlelere ulaşmasında önemli bir rol oynamaktadır [3].

Mühendislik eğitimi ve teknik eğitimde, öğrencilere verilecek teorik bilgiler yanında, uygulama çalışmalarının ve deneylerin kapsam dışında tutulması düşünülemez. Mühendislik eğitimi ve teknik eğitim öğrencileri için pratik deneyim sağlamanın klasik yolu, laboratuvar temelli sistemler kullanmaktır [5].

Uzaktan eğitimin bütün bilim dallarında yaygınlaşması ve başarılı olması için eğitim ve öğretimin vazgeçilmez unsurları olan laboratuvarlarında uzaktan erişilebilir olması ve laboratuvar deneylerinin uzaktan yapılabilmesi, kısaca öğrencilerin laboratuvar imkânlarına sadece derslerde ve uygulama saatlerinde değil her zaman ve her yerden erişebilmeleri önemli bir aşama olacaktır. Bu gereksinim sanal laboratuvar tanımını ortaya çıkarmıştır [6].

Özellikle internet altyapısı ile birlikte web tabanlı yazılımların ve bu yazılımları destekleyen bilgisayarların büyük bir hızla gelişmesi, internet üzerinden ses görüntü veri aktarım hızlarının her geçen gün daha da artmasını ve uzaktan eğitimin yaygınlaşmasını sağladığı gibi, laboratuvar uygulamalarının internet üzerinden

yapılabilmesini kolaylařtırmakta ve web tabanlı laboratuvarların yaygınlařmasını saęlamaktadır [7].

Bilgisayar bilimlerine iliřkin eęitim verilen bilgisayar blmlerinde, bilgisayar donanımını ęretmeye ynelik olarak bilgisayar sistemleri, bilgisayar donanımı ya da bilgisayar yapısı isimli dersler bulunmakta ve bu derslerde mikroiřlemcinin ęretimi nemli ve kapsamlı yer tutmaktadır [8]. Konunun ęretimi sırasında her komut ve adresleme modlarının ęrenilmesi iin ęrenciden beklenen temel bilgi dzeyine ulařması uzun sreli deneysel alıřma ile saęlanabilmektedir. Ayrıca deneysel alıřma sırasında mikroiřlemci ierisinde gerekleřen olayların izlenebilmesi mmkn deęildir. Bu eksiklikleri giderebilmek iin konunun animasyon ya da simlasyonlar ile ęretimi nemli katkı saęlar [9].

Mikroiřlemci kavramı zellikle Batılı lkelerde ok nceden gndemleri iřgal etmiřtir. rneęin 1978 Haziranında İngiltere Bařbakanı, mikroiřlemcilerin kullanıma girmesinin belirli sanayi dallarındaki istihdam modeli zerinde yaratacaęı etkiden duyduęu kaygıyı dile getirmesine karřılık, Sanayi Bakanlıęı ve Ulusal Giriřim Dairesi, mikroiřlemci teknolojisini kullanmayı ve geliřtirmeyi zendirecek bir giriřim gerekleřtirmiřtir. Mikroelektronik ara ve gerelerin, arařtırılması, geliřtirilmesi ve retimi iin personel eęitimi ve ęretimi ile bunların dięer sanayi dallarına uygulanması da, byle bir geliřtirme programının nemli bir yn olarak grlmřtr [10].

Mikroiřlemcili sistemlerin endstrinin her dalında istihdam modeli oluřturmasının farkına varan eęitim yneticileri, mikroiřlemcilerin alıřmasını ve programlanmasını ęretmenin veya ęrenmenin birinci yolunun mikroiřlemci eęitim setlerinin kullanımı olduęunu grmřlerdir. Trkiye'nin bu konudaki ilk alıřması; Milli Eęitim Bakanlıęının Dnya Bankası ile birlikte gerekleřtirdięi Mesleki Eęitim Projesi adı altında, bnyesinde elektronik blm bulunan meslek liselerine eęitim amalı mikroiřlemci setlerinin alınması olmuřtur [11].

Aynı kapsamda, 1994 yılında Milli Eęitim Bakanlıęı tarafından yrtlen Endstriyel Okullar Projesiyle, eřitli meslek alanlarında ihtiya duyulan 42 adet

yabancı teknik ders kitabının tercüme hakları satın alınmıştır. Bu proje kapsamında mikroişlemci konusunu anlatan, yararı inkar edilemeyecek önemli miktarda kitap tercümesi yapılmıştır [12].

Bu süreç, 1997 yılında YÖK'ün Dünya Bankası ile yaptığı Endüstriyel Eğitim Projesi kapsamında, Teknik Eğitim Fakültelerinin Elektronik ve Bilgisayar Eğitimi Bölümlerine ve Meslek Yüksek Okullarının Elektronik ve Bilgisayar donanımıyla ilgili bölümlerine eğitim amaçlı mikroişlemci deney setleri alınmasıyla devam etmiştir [13].

Günümüzde mikroişlemci ve mikrodenetleyici eğitimi iki farklı yöntemle verilmektedir: Bilgisayar ve multimedya araçları yaygın kullanımda değilken başlayan ve hali hazırda uygulanmakta olan yöntemde Mikroişlemci Eğitim Setleri kullanılarak eğitim verilmektedir. Bu yöntemde; plaket üzerine mikroşlemci / mikrodenetleyici, göstergeler ve tuş takımlarını barındıracak şekilde devre elemanlarının monte edildiği setler kullanılmaktadır. İkinci yöntemde ise; günümüzde bilgisayar sistemlerinin yaygınlaşmasıyla birlikte her alanda kullanımı yaygınlaşmış olan bilgisayar simülasyonu/animasyonu kullanılmaktadır. Bilgisayar simülasyonu, gerçekleştirilmesi uzun zaman alan ve yüksek maliyet gerektiren sistemlerin çalışmalarının bilgisayarlar yardımıyla taklit edilmesidir. Bu taklit (simülasyon) gerçek sistemin bütün fonksiyonlarını kapsamakta ve kullanıcıya sistemin çalışması hakkında bilgi vermektedir.

Mikroşlemci eğitim setleri, öğrencilerin laboratuvar ortamında kuramsal derslerde gördükleri konulara göre aldıkları bilgileri uygulama imkanı bulmaktadırlar. Fakat laboratuvarlarda geleneksel (elektromekanik) eğitim setlerinin kullanımı aşağıda belirtilen zorlukları ortaya çıkartmaktadır [14]:

- Donanımsal olduğundan pahalıdır,
- Modül eklenmesi zor ve güncellenme imkanı yoktur,
- Bir mekandan diğerine taşınması zordur,

- Elektronik ve mekanik parçalardan meydana geldiğinden sık sık arıza yapar, bakım ve onarım gerektirir ve bundan dolayı teknisyen ihtiyacı vardır. Bu da ek maliyet demektir,
- Sınıfların kalabalık olmasından dolayı, her bir öğrenciye tek bir eğitim seti düşmediğinden öğretim açısından olumsuz sonuçlar doğurur,
- Yazılan programa göre sette gerçekleştirilen olayların (veri alış / verışı) gözlenmesi zordur,
- Yazılan programın makine kodlarının tek tek sete girilmesi oldukça zordur,
- Programda düzeltme ve geriye dönüş maharet gerektirir,
- Kısıtlı tuş takımına sahip setlerde program denetimi zordur.

Bilgisayar simülasyonu, internet ortamına aktarılarak öğrencinin kullanımına açılabilmesi yanında CD benzeri saklama ortamlarında öğrencinin kişisel bilgisayarda kullanmasına yönelik kopyalanabilmekte ve dağıtılabilmektedir. Bilgisayar simülasyonu, öğrencilerin bilgisayar başında kuramsal derslerde gördükleri konulara göre aldıkları bilgileri uygulama imkanı bulacakları bir ortam sağlamaktadır. Geleneksel eğitim setlerinin kullanılması yerine, maliyet verimliliği, kullanılabilirlik, aktif öğrenme, güvenlik, idari faydalar ve ilave modüllerle güncelliğini koruyabilen yazılım tabanlı mikroişlemci/mikrodenetleyici eğitim setlerinin kullanımı ülkemiz ve dünya ekonomisine katkıda bulunmaktadır.

Yapılan çalışmada, mikroişlemci/mikrodenetleyici simülatörünün tam bir öğretim/öğrenim aracı olarak tasarlanması ve gerçekleştirilmesi için birçok bilimsel kaynak taranmıştır. Özellikle daha önce yapılan çalışmalarda sonuç ve öneriler dikkate alınarak çalışmalarda eksiklik ve olumsuzluklar belirlenmiş ve tasarımı amaçlanan simülatörlerde giderilmeye çalışılmıştır.

Ülkemizde benzer konularda, Windows 3.1 işletim sistemi çıkıncaya kadar olan zaman diliminde yapılan çalışmalar DOS işletim sistemi üzerinde çalışmak üzere tasarlanmış ve görsellikten uzaktır. Windows 3.1 işletim sisteminin çıktığı yıldan günümüze kadar yapılan çalışmalarda görsel yönden zenginleştirme yapılmış olsa da yapılan çalışmaların çoğunda mikroişlemcinin içerisinde gerçekleşen olaylar saat çevrimi bazında adımlanarak gösterilmemiş, sadece kaydedici ve bellek



birimlerindeki son durumlar gösterilmiştir. Gene bu çalışmaların büyük çoğunluğunda mikroişlemcinin çevre birimleriyle çalışması simüle edilmemiştir.

Nartkaya tarafından yapılan “Intel 8085 Mikroişlemcisinin Simülasyonu” adlı çalışma, Windows ortamına uygun görsellikte hazırlanmıştır ancak simülatörün gerçekleştirilmesinde en önemli etkenlerden birisi olan denetlemede herhangi bir çevre biriminin (LED, anahtar, trafik lambaları gibi) denetimi göz önüne alınmamıştır [12].

John D. Carpinelli ve Fabio Jaramillo tarafından yapılan “Dijital tasarım, bilgisayar organizasyonu ve mimarisi için simülasyon araçları” adlı çalışma üç simülatörden oluşmaktadır. Relatively Simple Computer System simülatörü bir mikroişlemci, hafıza ve bir giriş/çıkış aygıtından oluşan bir bilgisayarın davranışını göstermektedir. The Relatively Simple CPU simülatörü bir mikroişlemcinin iç fonksiyonlarını incelemektedir. Java ile hazırlanan bu çalışmada hata ayıklama ile ilgili hiçbir eklenti bulunmamaktadır. Aynı şekilde çevre birimlerinin denetimi göz önüne alınmamıştır [15].

Alfredo del Rio ve arkadaşları tarafından yapılan “Bilgisayar destekli mikrodenetleyicilerin öğrenilmesi” adlı çalışma 8051 mikrodenetleyicisi için hazırlanmıştır. UVI51 adlı simülatör CPU ve bağlı çevrebirimlerinin benzetimi, bağlı çevrebirimlerinin durum ve konfigürasyonlarını gösteren grafiksel pencereler ve dış çevrebirimleri ile etkileşimin simüle edildiği parçalardan oluşmaktadır [16].

Kurat tarafından yapılan “Internet Tabanlı PIC16F84 Eğitimi” adlı çalışmada her komut için birer animasyon tasarlanmıştır. Bu animasyonlara kullanıcı komutun parametrelerini belirlenmiş kutulara girmekte animasyon butonuna basarak izlemektedir. Bu çalışma bahsettiğimiz anlamda bir simülatör değildir [17].

Topaloğlu tarafından yapılan “PC Tabanlı Fonksiyonel Mikroişlemci Simülatör Tasarımı ve Gerçekleştirilmesi” adlı çalışmada 6502 mikroişlemci simülatörü beş modülden meydana getirilmiştir. Bu modüller editör, assembler (çevirici), debugger (hata ayıklayıcı), animatör ve sanal uygulamalardır. Editör kısmında yazılacak

assembly dilindeki program bir butona basılarak derlenmektedir. Derlenen program hem mikroişlemci içerisinde gerçekleşen olayları adım-adım çalıştırabilmekte hem de çevre birimleriyle çalışmayı simüle edebilmektedir [14]. Tutulan program günlüğü ile her işletilen komutu ve kaydedici içeriklerinin aldığı durum takip edilebilmektedir.

Taşkın tarafından yapılan “MC6811 Mikrokontrolücüsü Simülasyon Araçları” adlı çalışma kaynak kodundan nesne kodu elde etmeye yarayan bir derleyici ve derlenmiş kaynak kodunu çalıştırmaya yarayan bir simülatörden oluşmaktadır. Grafik bir arayüz sayesinde mikrokontrolcünün bütün işlevleri, CPU, veri ve adres yolları, portları görülebilmektedir. Çalışmada çevre birimlerinin denetimi göz önüne alınmamıştır [18].

Literatürde incelenen tüm bu çalışmalar PC tabanlı kişisel bilgisayarda çalışmaktadır. Bu çalışmada ise hazırlanan 8085 ve PIC16F84 simülatörleri web tabanlıdır ve uzaktan eğitime katkı sağlamaktadır. Bunun yanında komutların mikroişlemci/mikrodenetleyici iç mimarisinde çalışması en küçük pals biriminden adımlarla gösterilmektedir. Komutların sadece kaydedicilere etkisi değil bellek üzerinde gerçekleştirdiği işlemlerde kayıtlanmakta ve görüntülenmektedir.

Bu çalışma, mikroişlemcinin en temel unsurlarının dahil edildiği, mikroişlemci/mikrodenetleyici simülatörleri, fakülte ve yüksek okulların dışında tüm meslek liselerinin elektronik ve bilgisayar bölümlerinde okutulmakta olan mikroişlemciler dersleri için, üstün grafiksel ve görsel özelliklere sahip ve uzaktan eğitimde kullanılmak üzere Macromedia Flash programı ile gerçekleştirilmiş bir tasarımıdır.

Bu tez yedi bölümden meydana gelmiştir: Giriş olarak verilen birinci bölüm de tezin oluşmasına etki eden fikirler oluşturulmakta, tezin ortaya çıkmasına neden olan eksiklikler ortaya konulmakta ve tez düzeni açıklanmaktadır. İkinci bölümde, mikroişlemci simülatöründe ele alınan 8085 işlemcisi ve genel işlemci kavramlarını kapsamaktadır. Bu bölümde mikroişlemcilerin tarihçesi ve genel yapısı, 8085 mikroişlemcisinin ayak bağlantıları, aritmetik ve mantık arbirimi, kaydediciler,

zamanlama ve kontrol birimi, komut kaydedici ve komut çözücü devreleri, kesme seri giriş-çıkış kontrol devreleri, adresleme modları ve assembly komut seti ele alınmaktadır.

Üçüncü bölümde, mikrodnetleyici kavramı, PIC16F84 mikrodnetleyicisinin içyapısı, hafıza organizasyonları, özel dosya kaydedicileri, I/O portları, zamanlama ve kontrol, adresleme yöntemleri ve kesme konuları ele alınmaktadır.

Dördüncü bölümde, simülasyon kavramı, sanal laboratuvar ve yararları, uygulamayı gerçekleştirdiğimiz araç olan Macromedia Flash programının (artı ve eksi yönleriyle) tanıtımı ve mikroişlemci/mikrodnetleyicilerde simülasyon kavramı ele alınmaktadır.

Beş ve altıncı bölümlerde, 8085 mikroişlemci/ PIC16F84 mikrodnetleyici için komutlarının yazıldığı ve derleme işleminin yapıldığı editör, derlenen programın hatalardan arındırılması için hata ayıklamada kullanılan kaydedici ve bellek içeriklerinin görüntülediği pencereler, komutların mikroişlemci/mikrodnetleyici iç mimarisinde icrasının simüle edildiği animatör ve modüler yapıdaki çevre birimleriyle çalışmasının gözlemlenebileceği deneyler penceresi detaylandırılmaktadır.

Yedinci bölümde ise sonuç ve öneriler verilmiştir.

## BÖLÜM 2. INTEL 8085 MİKROİŞLEMCİLER

### 2.1. Mikroişlemci Kavramı

Günümüzde kullanılan bilgisayarların özelliklerinden bahsedilirken duyduğunuz 80386, 80486, Pentium-I, Pentium-II, Pentium-III, Pentium-IV isimleriyle anılan elemanlar birer mikroşlemcidir (microprocessor). Mikroşlemciler bilgisayar programlarının yapmak istediği tüm işlemleri yerine getirdiği için, çoğu zaman merkezi işlem ünitesi (CPU-Central Processing Unit) olarak adlandırılır [19].

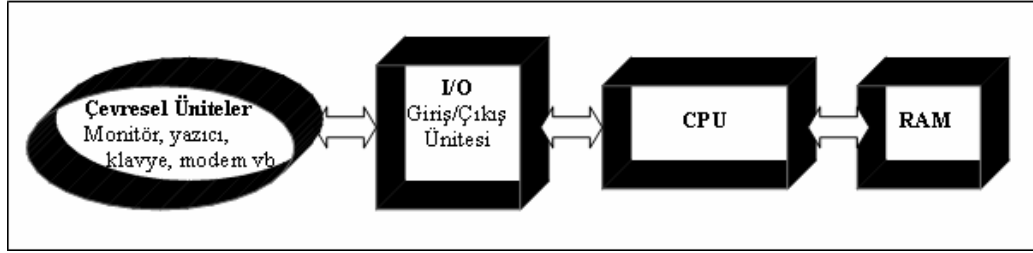
Mikroşlemci, makine dilindeki komutları yorumlayarak gerekli işlemlerin yerine getirilmesi için denetiminde bulunan bir dizi fonksiyonel elemanı yetkilendiren ve bu operasyon sonucunda elde edilen verilere göre bağlı bulunduğu sistemi çalıştıran bir elemandır. Mikroşlemci girdileri alarak verilen ölçütlere göre çok kısa sürede, gerçek zamanlı cevap verebilen bir sistemdir [14].

PC adını verdiğimiz kişisel bilgisayarlarda kullanıldığı gibi, bilgisayarla kontrol edilen sanayi tezgahlarında ve ev aygıtlarda da kullanılabilir. Bir mikroşlemcinin işlevini yerine getirebilmesi için aşağıdaki yardımcı elemanlara ihtiyaç duyar. Bunlar:

- Input (Giriş) ünitesi
- Output (Çıkış) ünitesi
- Memory (Bellek) ünitesi

Bu üniteler CPU entegresi dışında, bilgisayarın ana kartı üzerinde bir yerde farklı entegrelerden veya elektronik elemanlardan oluşur. Aralarındaki iletişim ise veri yolu (data bus) ve adres yolu (adres bus) olarak isimlendirilen iletim hatları

üzerinden yapılır.



Şekil 2.1 Bir mikro işlemci sisteminin temel bileşenlerinin blok diyagramı

Intel, Cyrex, AMD, Motorola mikroişlemci üreticilerinden bir kaçıdır. Günümüzde mikroişlemciler genellikle PC adını verdiğimiz kişisel bilgisayarlarda kullanılmaktadır [19].

Mikroişlemcinin özellikleri şu şekilde özetlenebilir:

- Mikroişlemci, tek entegre halindeki bir elektronik işlem elemanıdır.
- İlk olarak 1970’de hesap makinelerinde kullanılmak üzere üretilmiştir.
- Bugün en çok kullanılma yeri mikrobilgisayarlardır.
- Bilgisayardaki bütün işlemler mikroişlemci tarafından yürütülür.
- Mikroişlemciler için, bilgisayarın beyni tanımlaması yapılmıştır. Bu nedenle, mikroişlemciler bilgisayar ile birlikte incelenir.
- Mikroişlemcilerden elektronik kontrol devrelerinde de yararlanır.

Mikroişlemcilerin ayrımı aynı anda işleyebildiği bit sayısına göre yapılmaktadır. Günümüze kadar 4, 8, 16, 32 ve 64 bitlik mikroişlemciler üretilmiştir [20].

Günümüz mikroişlemcilerinde öne çıkan bazı üstün özellikler aşağıda sıralanmıştır:

- 0.09mikronluk silikon yüzey birleşimi,
- Komutların hızlı işlenmesini sağlayan farklı birimlerde işlenmesini tanımlayan süper ölçekli mimari yapının geliştirilmesi,
- Programda daha sonra çalıştırılacak komut grubunun tahmin edilerek ona göre hazırlık yapılmasını sağlayan dallanma tahmin yeteneğinin kazanılması,

- Yüksek performanslı tam sayı ve kayan nokta birimlerinin geliştirilmesi,
- 128 bit veri yolu ve 32 bit adres yollarının gerçekleştirilmesi,
- İşlemci güç yönetim biriminin geliştirilmesi,
- Tek komutla birden çok komutun işlenmesini sağlayan tekniğin (SIMD- Single Instruction Multiple Data) oluşturulması,
- Birden fazla komutun bir çok iş hattında işlenmesini sağlayan derin iş hattı teknolojisinin geliştirilmesi,
- 533 MHz'lik sistem veri yollarının gerçekleştirilmesi.

Yukarıda sıralanan birçok özellik sayesinde günümüz mikroişlemcileri, bilgisayar sistemlerinin çok yüksek performansa çıkmalarını sağlamıştır. Bu sebepten çok ileri tarihlerde yapılması düşünülen veya planlanan araştırma ve keşifler daha yakın tarihe çekilebilmiştir [14].

Uygulama alanları: Mikroişlemcinin en önemli parça olduğu, hayatın her yerinde uygulama alanı bulan bilgisayar sistemleri muhasebe, bilim ve mühendislik dallarında, endüstrinin gerçek zamanlı kontrol uygulamalarında ve bilgisayar destekli eğitim alanlarında çok sık kullanılmaktadır.

## 2.2. Mikroişlemcilerin Tarihçesi

Elektronik bilgisayar düşüncesi, 1919 yılında Eccles ve Jordan'un "flip-flop devresini bulması ile ortaya atıldı. Pennsylvania Üniversitesinde 1942 yılında başlatılan bir çalışma 1945 yılında sonuçlandı ve ilk modern bilgisayar olarak bilinen ENIAC "Electronic Numerical Integrator and Calculator" ortaya çıktı. Dr. Von Neumann ve arkadaşları programı bellekte saklanabilen ilk bilgisayar olan EDVAC (Electronic Discrete Variable Automatic Computer)'ı ortaya çıkardı. 1947 yılında 23.000 röle ve 13.000 tüpten oluşan SSEC (Selective Sequence Electronic Calculator) IBM firmasınınca gerçekleştirildi [21].

1948 yılında transistörün bulunuşu, bilgisayar teknolojisinin gelişmesinde sıçramaya neden olmuştur. 1960'lı yıllara gelindiğinde, artık bilgisayarlar standart olarak üretilip satılmakta veya kiraya verilmekteydi [21].

Yarıiletken teknolojisindeki olağanüstü hızlı gelişimin sonucu olarak, 1971 yılında ilk tek tüm devre içine sığdırılmış MIB (Merkezi İşlem Birimi) Intel firması tarafından piyasaya sürüldü. 4 bitlik olan bu ilk mikroişlemcinin adı I-4004 idi. Intel 8080, I4004, I4040 ve I8008'in arındandan üretilmiştir ve 8 bitlik mikroişlemcilerin ilkidir ve 1974'te kullanıma sunulmuştur. +5, -5 ve +12V'luk üç gerilim kaynağı gerektiren I8080, yerini 1976 yılında tek 5V'luk kaynak kullanan I8085'e bırakmıştır. I8085 yazılım olarak I8080 ile uyumlu olmakla beraber, donanımları farklıdır. I8080'de adres ve veri yolları için bağımsız ayaklar bulunurken, I8085'te veri yolları ile adres yolları paylaşımlı olarak düzenlenmiştir [21].

Sekiz bitlik mikroişlemciler: Sekiz bitlik mikroişlemciler yaygın olarak ve en uzun süre kullanılan mikroişlemcilerdir. Bugün bilgisayarda kullanım bakımından devrini tamamlamış olmakla beraber, halende özel kumanda ve kontrol sistemlerinde kullanılmakta ve eğitim amaçlı çalışmalarda bu tür mikroişlemcilerden yararlanılmaktadır. 8 bitlik mikroişlemciler ikinci nesil mikroişlemciler olarak tanınırlar. En çok kullanılan 8-bitlik mikroişlemciler Tablo 2.1'de verilmektedir [20].

Tablo 2.1 En çok kullanılan 8-bitlik mikroişlemciler

Yapımcı Firma	Üretim Tarihi	Mikroişlemci Kod No	Çalışma Hızı
INTEL	1973	8080	0,5 MHz
INTEL	1976	8085	1,4 MHz
INTEL	1976	8085A	3-125 MHz
INTEL	1976	8048	0,4 MHz
ZILOG	1974	Z80	2,5 MHz
ZILOG	1974	Z80A	4 MHz
MOTOROLA	1974	6800	2 MHz
MOTOROLA	1975	6802	1-2 MHz
MOTOROLA	1979	6809	4 MHz
ROCKWELL	1974	6502	2 MHz

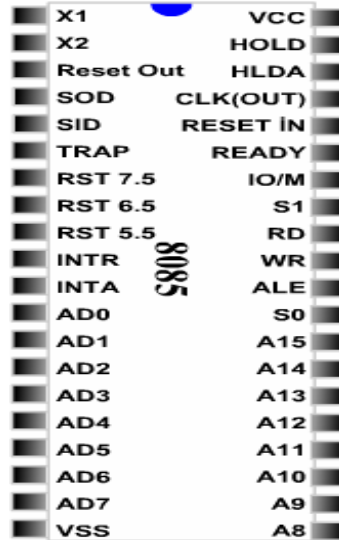
Tablo 2.1'de verilen mikroişlemciler 40 pinli olup, 8 bit veriyolu, 16 bit adres yolu ve 8 / 16 bit kaydedicilere sahiptir.

Mikroelektronik teknolojisinin hızla gelişmesi beraberinde mikroişlemcilerin

gelişmesini dolayısıyla da sistemde kullanılan program ve yazılımların da gelişmesini sağlamıştır. Yeni geliştirilen bir mikroişlemci anında mikrobilgisayarlarda kullanılmış ve buna göre de kısa sürede bir çok yazılımlar geliştirilmiştir. Yazılımların kapladığı bellek alanlarının büyümesi beraberinde bellek problemini doğurmuş ve mikroişlemcilerde yeni tasarımların oluşmasına yol açmıştır. Bunlardan en önemlisi, gerçek 16-bit, 32-bit ve 64-bitlik mikroişlemcilerin ortaya çıkması ve işlemcilerin performansını artıran destek devrelerinin geliştirilmesidir [14].

### 2.3. Intel 8085 Mikroişlemcisinin Özellikleri

Intel 8085 mikroişlemcisi, veri yolunun 8 bit genişliğinde olması ve aritmetik - mantık birimlerinin 8-bit kelimeler üzerinde işlem yapmak için tasarlanması nedeni ile 8-bit mikroişlemcidir. 8085 mikroişlemcisi 40 bacaklı (pinli) çift hatlı (DIN) entegre yapısında olup, mikroişlemcide adres, veri, besleme ve kontrol sinyalleri bulunur (Şekil 2.2).



Şekil 2.2 Intel 8085 mikroişlemcisinde bulunan pinler

Bütün sinyallerin TTL uyumlu olduğu 8085 mikroişlemcisi, +5V besleme gerilimi ile çalışır. Mikroişlemcinin 16-bitlik adres yoluna sahip olması nedeni ile, adreslenebilecek maksimum bellek bölgesi 64 KBayttır. 8085 Mikroişlemcisi, 3



MHz'lik tetikleme sinyali ile çalışırken, 8085-2 mikroişlemcisi 5 MHz tetikleme sinyali ile çalışır [22].

Tablo 2.2 8085A mikroişlemcisinde bulunan pinlerin işlevleri

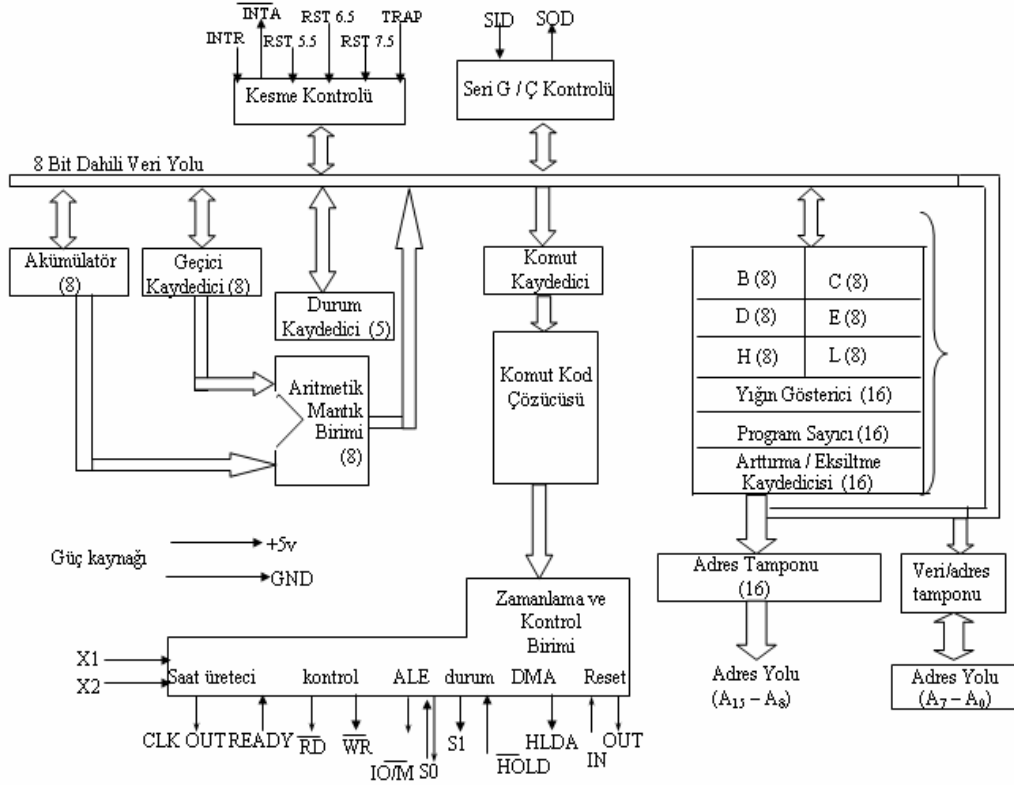
B. No	Bacak Adı	Açıklama	No	Bacak Adı	Açıklama
1	X1	Giriş	21	A8	Adres yoluna çıkış
2	X2	Giriş	22	A9	Adres yoluna çıkış
3	RESET OUT	Çıkış	23	A10	Adres yoluna çıkış
4	SOD	Çıkış	24	A11	Adres yoluna çıkış
5	SID	Giriş	25	A12	Adres yoluna çıkış
6	TRAP	Kesme girişi	26	A13	Adres yoluna çıkış
7	RST 7.5	Öncelikli kesme	27	A14	Adres yoluna çıkış
8	RST 6.5	Öncelikli kesme	28	A15	Adres yoluna çıkış
9	RST 5.5	Öncelikli kesme	29	S0	Komut kod çözücüsü
10	INTR	Kesme isteği	30	ALE	Adres kilidi işlemcisi
11	NOT INTA	Kesme bilgisi	31	WR	3 durumlu yaz çıkışı
12	AD0	Adres/veri yoluna G/Ç	32	RD	3 durumlu oku çıkışı
13	AD1	Adres/veri yoluna G/Ç	33	S1	Komut kod çözücüsü
14	AD2	Adres/veri yoluna G/Ç	34	I0 / M	Oku/yaz çıkış yönlendir.
15	AD3	Adres/veri yoluna G/Ç	35	READY	Hazır olma durumu
16	AD4	Adres/veri yoluna G/Ç	36	(RESET IN)	Kesme girişi
17	AD5	Adres/veri yoluna G/Ç	37	CLK OUT	Saat sinyali çıkışı
18	AD6	Adres/veri yoluna G/Ç	38	HLDA	Tutma bilgisi
19	AD7	Adres/veri yoluna G/Ç	39	HOLD	Tutma girişi
20	VSS	Toprak	40	VCC	+5v besleme girişi

#### 2.4. Intel 8085 Mikroişlemcisi İç Yapısı

Mikroişlemcinin genel yapısının açıklanması sırasında kullanılan birimler 5 grup altında incelenebilir:

1. Aritmetik - Mantık birimi (ALU),
2. Kaydedici dizisi,
3. Zamanlama ve kontrol birimi,
4. Komut kaydedici ve komut kod çözücü devreleri,
5. Kesme ve seri giriş/çıkış kontrolü devreleri.

8085 mikroişlemcisinde bulunan birimlerin ve birimler arasında iletişimi gösteren hatların detaylandırılması ile Şekil 2.3'deki blok şema elde edilir.



Şekil 2.3 8085 mikroişlemcisi işlevsel blok şeması (Ekiz 2005)

#### 2.4.1. Aritmetik - mantık birimi

8085 mikroişlemcisi 8-bitlik mikroişlemci olduğundan, ALU biriminde bulunan devreler 8-bitlik kelimeler üzerinde işlem yapacak şekilde tasarlanmışlardır. ALU biriminde; ikili sayı '1' artırabilir, '1' eksiltilebilir veya iki adet 8-bitlik sayı üzerinde VE, VEYA, ÖZEL VEYA, toplama, çıkarma, karşılaştırma işlemleri yapılabilir. Akümülatör, geçici kaydediciler, durum kaydedicisi ve onluğa ayarlama devreleri aritmetik - mantık birimi ile ilgili devreler olarak isimlendirilir.

**Akümlatör:** ALU tarafından üzerinde işlem yapılacak sayılardan birini tutan ve ALU tarafından yapılan işlemin sonucunu saklayan 8-bitlik bir kaydedicidir [12].

**Onluğa Ayarlama Devresi:** BCD toplama veya çıkarma işleminde, akümülatörü

onluğa ayarlama devresi kullanılır. BCD formunda yapılan toplama işleminde toplam 9'dan büyükse, sonuca +6 sayısı eklenerek düzeltme yapılır.

Durum Kaydedicisi: Her aritmetik veya mantık komutunun yürütülmesinden sonra, durum kaydedicisinde bulunan beş durum bayrağı işlem sonucunda oluşan durumları belirtmek üzere '1' veya '0' yapılır (Şekil 2.4). Durum bayraklarının herhangi birisi (özellikle Z veya C), mikroişlemciye bir başka programa veya program kısmına dallanmasını veya atlamasını söylemek için kullanılabilir.

Durum Kaydedicisi							
S	Z	X	AC	X	P	X	C

Şekil 2.4 8085A Durum kaydedici formatı

Elde bayrak biti (Carry flag – C): Toplama işlemi sonucu FF değerinden büyük çıkarsa elde bayrağı 1, değilse 0 yapılır. Çıkartma işlemi sonucu değer negatif çıkarsa C biti ödünç bayrağı gibi kullanılır ve 1 yapılır.

Eşlik bayrak biti (Parity flag – P): Akümülatörün içindeki sayıda bulunan birler toplamı çift ise 1, değilse 0 yapılır.

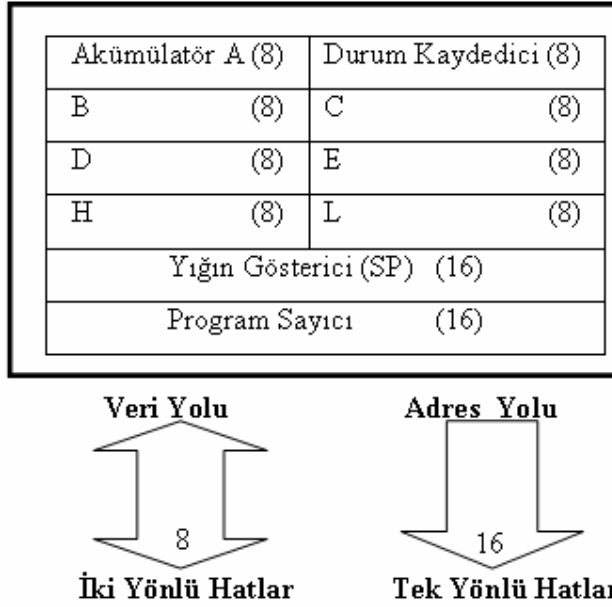
Yardımcı elde bayrak biti (Auxiliary Carry flag - AC): Akümülatörde işlenen bilginin 3. bitinden elde değeri oluşursa '1' yapılır. Bu bayrak BCD toplama veya çıkarma yapılırken, onluğa ayarlama işleminin yapılması gerektiğini belirtmek için kullanılır.

Sıfır (0) bayrak biti (Zero flag - Z): Son işlem sonucunun 0 olması durumunda 1 yapılır.

İşaret bayrağı biti (Sign flag - S): Bir işlem sonrasında, akümülatörde oluşan verinin en yüksek bitine bakarak değer alır. En yüksek bit 1 ise S bayrağı 1 yapılır. Ters durumda S bayrağı 0 yapılır. Sayı en yüksek biti 1 ise negatif 0 ise pozitifdir.

### 2.4.2. Kaydedici dizisi

Intel 8085 mikroişlemcisinde 10 adet kaydedici bulunur (Şekil 2.5). Bu kaydedicilerden bir kısmı programcı tarafından kullanılabilir şekilde genel amaçlı iken, bir kısmı yalnızca mikroişlemci tarafından programların işlenmesi sırasında kullanılır.



Şekil 2.5 Intel 8085 mikroişlemcisinde bulunan kaydediciler

**Kaydedici Çiftleri:** 8-bitlik genel amaçlı kaydediciler tek kullanıldıkları gibi BC, DE ve HL biçiminde 16-bitlik kaydedici çifti olarak ta kullanılabilirler. Geçici kaydedici çifti W ile Z programcı tarafından kullanılmazlar.

**Yığın Göstericisi:** 16-bitlik yığın göstericisi, belleğin herhangi bir yerinde yığın kurabilmeye olanak sağlar. Alt programlara geri dönüş adresini saklamak için de kullanılır. Yığın bölgesinde aktif olan adresi gösterir. Yığın göstericinin değeri yığın bölgesine her bilgi yazılması durumunda 1 azalmakta ve bilgi okunması durumunda 1 artmaktadır. Değer değişimi, komutların işlenmesi sırasında mikroişlemci tarafından gerçekleştirilir.

**Program Sayacı:** Program sayacı 16-bit uzunluktadır ve işlenmekte olan komutun

bellek adresini gösterir. Program sayıcının içeriği, işlenen her komuttan sonra bellekteki bir sonraki komut veya verinin yerini gösterecek şekilde otomatik olarak 1 artırılır. Dallanma ve atlama komutları ile kullanılması durumunda, içeriği uygun değere kurularak programın yeni bellek bölgesindeki komutlar ile devam etmesi sağlanır.

Adres Tamponu: İki işlev görür. Program sayıcıdan, yığın göstericiden veya 16-bitlik kaydedici çiftlerinin birisinden gönderilecek adresin seçimini yapmak ve seçilen adresin adres hatlarında gerekli süre boyunca tutulmasını sağlamak. 8085 mikroişlemcisindeki 16-bitlik adres hattı mikroişlemcinin  $2^{16}$  yani 65536 bellek bölgesi adreslemesi yapmasına imkan tanır.

### **2.4.3. Zamanlama ve kontrol birimi**

Zamanlama ve kontrol biriminde bulunan devreler yardımı ile tüm mikroişlemci işlemlerinin senkronizesi sağlanır ve mikroişlemci ile çevrebirimleri arasında iletişim için gerekli kontrol sinyalleri üretilir.

Mikroişlemcinin çevre birimleri ile birlikte çalışmasını sağlayacak 'Clk Out', 'Ready', 'ALE', 'Hold', 'HLDA', 'Reset In', 'Reset Out' sinyalleri ile birlikte, veri yolu üzerindeki verinin şeklini gösteren 'RD' – 'WR' sinyalleri ve komut ile gerçekleştirilen işlemin türünü belirten  $S_1$  –  $S_2$  girişleri, zamanlama ve kontrol birimi içerisinde yer alır.

Kontrol birimi,  $X_1$  ve  $X_2$  girişlerine bağlanan kristal ile çalışır.

### **2.4.4. Komut kaydedici ve komut kod çözücüsü**

Komut kaydedici ve komut kod çözücüsü, komutun yorumlanması ve yapılan işlemin belirlenmesinde önemli bir yere sahiptir. Bir komut bellekten okunduğu zaman, veri yolu üzerindeki bilgi komut kaydedicisine yüklenir. Yüklenen bilgi, mikroişlemci tarafından yorumlanıp, komut ile gerçekleştirilmesi gerekli işlem bitirilinceye kadar komut kaydedicisinde tutulur.

Komut kod çözücü devre; komut kaydedicisinde tutulan komutu yorumlar ve komut ile yapılması gerekli işlemleri sıralayarak, işlemlerin yapılmasını sağlayacak uygun sinyalleri üretir.

Örneğin; bir programda 'ADD' komutu kod çözücüye gelince, kod çözücü komutu yorumlayarak bellekten veri getirmek gerektiğini belirler. Akümülatöre eklenecek sayıyı almak için 2. bir getirme işlemi gerektiğini bulur ve bunu kontrol birimine bildirir.

Toplama bitirildiğinde, kontrol devresi tarafından sonucun akümülatöre gönderileceği belirtilir. Kontrol devreleri program sayıcısının içeriğini bir sonraki bellek bölgesi adresini gösterecek şekilde artırır ve bellekten bir sonraki komutu okumak için yeni bir bellek okuma darbesi gönderir.

#### **2.4.5. Kesme ve seri giriş / çıkış kontrolü devreleri**

Mikroişlemcinin harici durum sinyalleri / kesmeleri ile uyumlu çalışması, kesme kontrolü devreleri üzerinden mikroişlemcinin ilgili birimlerine iletilir. 8085 mikroişlemcisinde, beş adet kesme girişi ve bir adet kesme bilgisi çıkışı bulunur.

8085 mikroişlemcisinin çevre birimleri ile bilgi paylaşımını sağlayan seri bilgi girişi (SID) ve seri veri çıkışı (SOD) sinyalleri, seri giriş/çıkış kontrolü devresinden gönderilir. Mikroişlemcinin çevre birimleri ile haberleşmesini sağlayan portlar ve harici olarak eklenen tamponlar, seri giriş/çıkış kontrolü devreleri içerisinde değerlendirilir.

#### **2.5. Intel 8085 Mikroişlemcilerinde Bulunan Kesmeler**

Kesme; mikroişlemcili sistemlerde çevre birimlerinden mikroişlemcinin kesme girişlerine gelen sinyal üzerine çalışmasını bir süre için kesip, bir başka işe yapması ve bu iş bitince de eski işine kaldığı yerden devam etmek olarak tanımlanabilir.

Günlük yaşamdan bir örnekle; ekmek kesmekte iken parmağımızı kesersek bu gibi

bir duruma alışık olup önemsemiyor (maskeleymiş) olabiliriz ya da önemsiyorsak ilk yapmamız gereken yaranın durumuna göre kanamayı durdurup tedavi işlemlerini halletmek daha sonrasında kaldığımız yerden ekmek kesmeye devam etmek olacaktır.

Örneğin; bir mikroişlemcili sisteminin büyük bir kağıt üretme makinesini kontrol ettiğini varsayalım. Mikroişlemcili sistemlerin görevlerinden birisi; buhar kazanının basınç emniyet sınırını aşmamasını kontrol etmektir. Eğer kazan basıncı emniyet sınırını aşarsa, mikroişlemcili sistem yakıtı derhal kesmeli ve basınç düşürme vanasını açmalıdır. Sistemden beklenen, mikroişlemcinin yaptığı işi kesmesi ve bu işlemi kazan patlamadan önce yapmasıdır [22].

8085A mikroişlemcisinin INTR, RST 5.5, RST 6.5, RST 7.5 ve TRAP olmak üzere beş tane kesmesi vardır.

INTR kesme girişini kullanabilmek için harici bir donanıma ihtiyaç vardır. INTR kesmesini kullanacak donanım mikroişlemcinin gideceği adresi adres yoluna koymak zorundadır. 8085A mikroişlemcisini RST 5.5, RST 6.5, RST 7.5 ve TRAP kesme girişleri bir RST komutu için gerekli harici donanım ihtiyacını ortadan kaldırır. Bu dört girişten herhangi birine bir kesme sinyali uygulanırsa ve bu girişler yetkilendirilmemişse ve maskelenmemişse; 8085A mikroişlemcisi, şu şekilde davranır. Aynı kesme sinyalinin kesmeye devam etmesini engellemek için kesmeler yetkisiz kılınır. Geri dönüş adresi yığına itilir. Program sayıcı o kesme girişine ait özel RST adresi ile yüklenir ve yürütme işlemi bu yüklenen adresten devam eder. Bu kesmelere ait dört adres; TRAP için 24H; RST 5.5 için 2CH; RST 6.5 için 34H ve RST 7.5 için 3CH dir. Her kesmeye hizmet verecek alt yordamın başlangıcı, bellekte karşılık gelen adrese konur [12].

Her komut çevriminin sonunda, 8085 mikroişlemcisi kesmelerinin yetkilendirilip-yetkilendirilmediği ve bir INTR kesmesi istenip istenmediği kontrol edilir. Bu iki koşul yerine getirilmişse, 8085 kesmeleri yetkisiz kılınır ve bir kesme alındı sinyali (INTA) gönderilir [22].

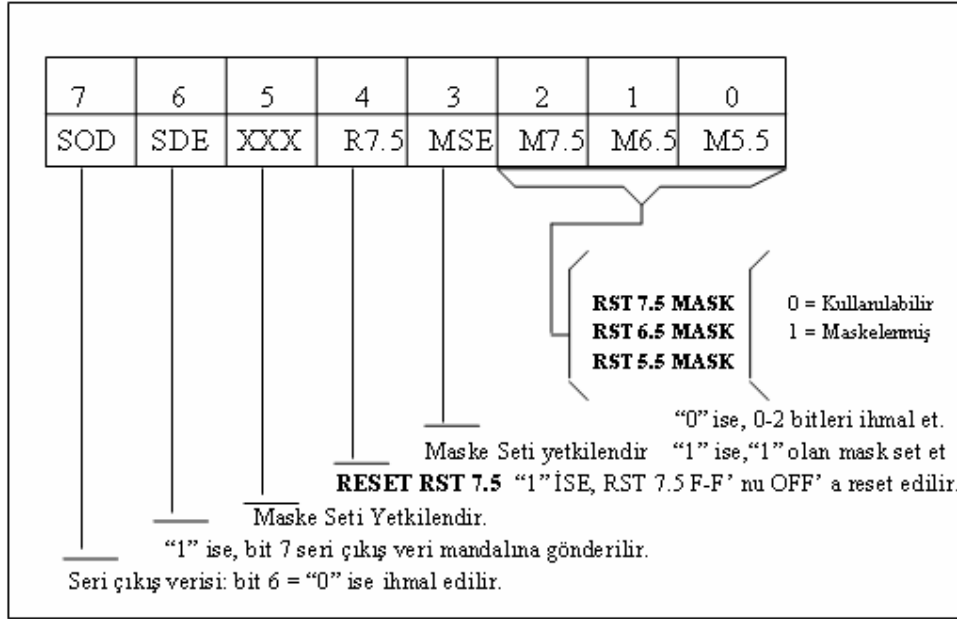
Kesme alt programlarında, TRAP dışında, kesmelerin tanınabilmesi için kesmeler EI komutuyla yetkilendirilmelidir. Bu çoğunlukla altprogramın sonuna doğru yapılır. Alt programın sonundaki bir RET komutu, geri dönüş adresini yığından çekerek yürütmeyi ana programa geri verir [12].

8085 mikroişlemcisinde bulunan beş kesme girişinin sabit bir hizmet önceliği vardır. Kesme sinyalleri eş zamanlı gelirse, öncelikli kesmeye hizmet verilir. Girişleri öncelikleri yukarıdan aşağıya olmak üzere; TRAP, RST 7.5, RST 6.5, RST 5.5 ve INTR şeklinde sıralıdır.

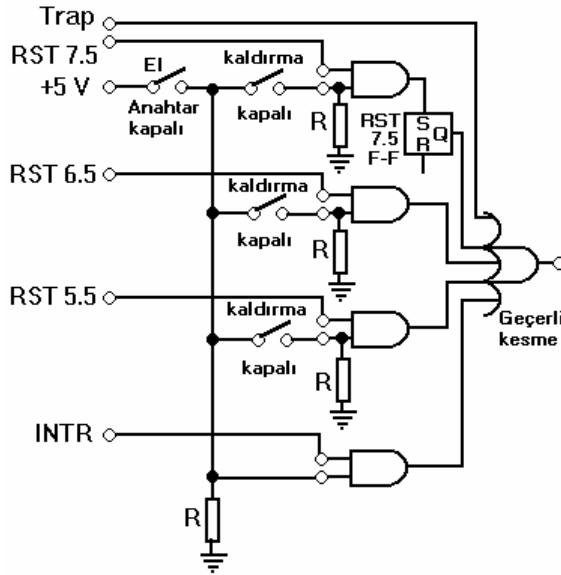
Maskeler: TRAP kesme girişi sadece en yüksek önceliğe sahip değildir, ayrıca hiçbir şekilde yetkisiz kılınmaz veya maskelenemez. 8085A mikroişlemcisi, TRAP'daki bir kesme isteğini her zaman tanır. Bu nedenle, bu giriş genellikle güç kesilmesi durumunda verileri çabucak saklamakta kullanılır. INTR girişi bir resetten sonra, bir kesmeyi yetkisiz kıl (disable interrupt: DI ) komutundan sonra veya kesme isteğinden sonra yetkisiz kılınır, EI komutuyla yetkilendirilir.

Bir resetten sonra, RST 7.5, RST 6.5 ve RST 5.5 kesme girişleri hem yetkisiz kılınır hem de maskelenir. Bu girişler EI komutuyla yetkilendirilir ve kesme maskesini kur (set interrupt mask: SIM) komutuyla istenilenlerin maskesi kaldırılabilir. Örneğin, maskeleme seçime bağlı olarak RST 7.5 ve RST 5.5' u yetkisiz kılmadan RST 6.5' u yetkisiz kılabilir. Şekil 2.6'da, SIM komutuyla birlikte kullanılan kelime için bit formatını göstermektedir. Eğer bu kelimenin 3. biti 1 ise, 0 ile 2 arasındaki bitler, seçime bağlı olarak sırasıyla RST 5.5, RST 6.5 ve RST 7.5 girişlerini maskeleyecek veya maskesini kaldıracaktır. Şekil 2.7'de beş 8085A kesme girişinin her birinin işlemci için geçerli bir kesme oluşturması için neyin gerektiğini anlamamıza yardımcı olarak gösterilmiştir.

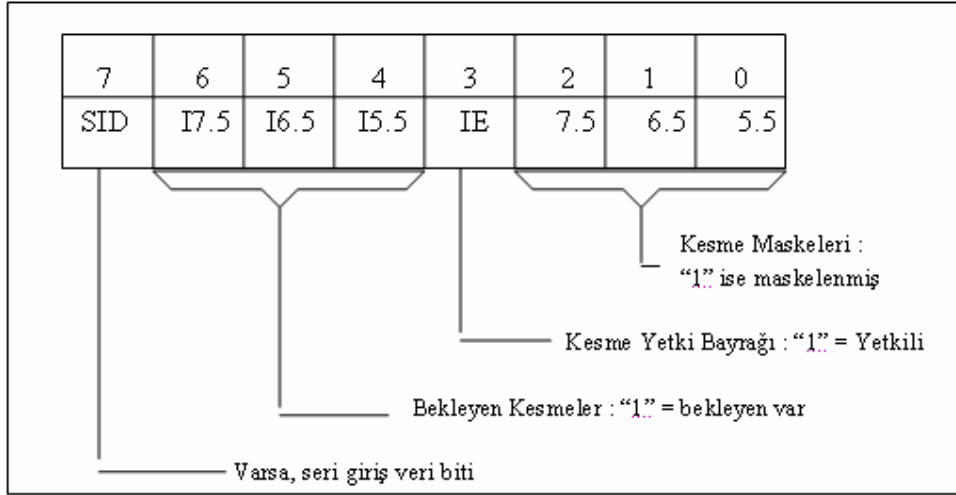




Şekil 2.6 SIM Komutu bit formatı (Nartkaya 1996)



Şekil 2.7 Kesme yetkilendirme ve maske kaldırma (Nartkaya 1996)



Şekil 2.8 RIM komut bit formatı (Nartkaya 1996)

## 2.6. Intel 8085 Mikroişlemcilerinde Adresleme Yöntemleri

8085 mikroişlemcilerine esneklik kazandırmak ve gerçekleştirilecek işleme göre uygun komutu kullanabilme imkanı sağlamak amacıyla, 5 farklı adresleme yöntemi kullanılmaktadır:

1. İvedi adresleme yöntemi (Immediate Addressing),
2. Doğrudan adresleme yöntemi (Direct Addressing),
3. Kaydedici adresleme yöntemi (Register addressing),
4. Kaydedici dolaylı adresleme yöntemi (Register Indirect Addressing),
5. İmalı adresleme yöntemi (Implied Addressing).

İvedi adresleme yöntemi: MİB içindeki kaydedici yada kaydedici çifti, boylarına uygun verilerle ivedi olarak yüklenebilir yada A kaydedici ile işleme sokulabilir.

Örnek komut: MVI A, veri (Komut ile verilen bir byte uzunluğundaki veri akümülatöre yüklenir).

Doğrudan adresleme yöntemi: Verinin bulunduğu adresin belirtilmesi şeklinde işlemlerin yürütülmesini sağlayan bir adresleme yöntemidir. Bu adresleme yöntemiyle geniş bir programlama olanağı bulunur ve program yazılışı kolaydır.

Terminaller ile bilgi iletişimi doğrudan adresleme yöntemi kullanılarak gerçekleştirilir [22].

Örnek komut: LDA Bellek (Bellekte bulunan veriyi akümülatör'e yükler).

Kaydedici adresleme yöntemi: Kaydediciler ve akümülatör arası veri transferi ile artırma ve eksiltme işlemleri ile bazı özel işlemlerde kullanılır. Kaydedici adresleme yöntemi hızlı çalışmayı sağlar ve kaydedici adresleme yöntemi ile yazılan komut, bellekte yalnızca bir bellek bölgesi yer kaplar.

Örnek komut: MOV A, B (B kaydedicisi içeriği A kaydedicisine aktarılır).

Dolaylı adresleme yöntemi: 8085 mikroişlemcisi, komut ile bir bellek bölgesi içeren kaydedicinin veya farklı bir bellek bölgesinin belirtildiği 'dolaylı adresleme yöntemi' destekler. Dolaylı adresleme yöntemini kullanan komutlarda, önce bir kaydedici çifti veya bellek bölgesi okunur ve okunan değer belirttiği bellek bölgesindeki veri üzerinde işlem yapılır. Bu durumda, işlem yapılacak bir verinin bulunduğu veya verinin gönderileceği adres dolaylı yoldan gösterilmek suretiyle gerekli işlem yürütüldüğü iki farklı dolaylı adresleme yöntemi oluşur:

1. Kaydedici dolaylı adresleme yöntemi: Bu yöntemde, işlem yapılacak verinin bulunduğu veya gönderileceği bellek bölgesi adresi, B-C, D-E veya H-L kaydedici çifti tarafından gösterilir.
2. Bellek dolaylı adresleme yöntemi: Bu yöntemde işlem yapılacak verinin bulunduğu adres bir başka bellek bölgesinde kayıtlıdır. Bu bölgenin gösterdiği adres kullanılarak veriye ulaşılır. Bu adresleme yöntemi daha çok, veri listeleme ve dosya formatlama uygulamaları için kullanışlıdır. 8080A ve 8085A mikroişlemcilerinde kullanılan dolaylı adresleme yöntemini kullanan komutlara örnek olarak aşağıdaki komutlar verilebilir:

Örnek komut: ANA bellek (Akümülatör ve bellekteki veriler arasında 've' işlemini uygular).

8085 mikroişlemcisinde, bellek bölgelerini belirlemek için HL kaydedici çifti kullanılır.

Örneğin; MOV M, A komutu ile daha önceden HL kaydedicisine kayıtlı bulunan bellek bölgesine Akümülatörün içeriği kopyalanır. Daha önceden yüklenen HL kaydedici çiftine bilgi yüklenirken, adresin yüksek değerli kısmı 'H' kaydedicisine düşük değerli kısmı ise 'L' kaydedicisine yerleştirilir.

İmalı adresleme yöntemi: Üzerinde işlem yapılacak kaydedici ima edilir. İmalı adresleme yöntemini kullanan komutların işlenen kısmında ima edilen kaydediciyi belirtmeye gerek yoktur.

Örnek komut: CMC (Durum kaydedicisindeki C bayrağının tersi alınır).

## 2.7. Programlanabilir Çevre Birimi 8255

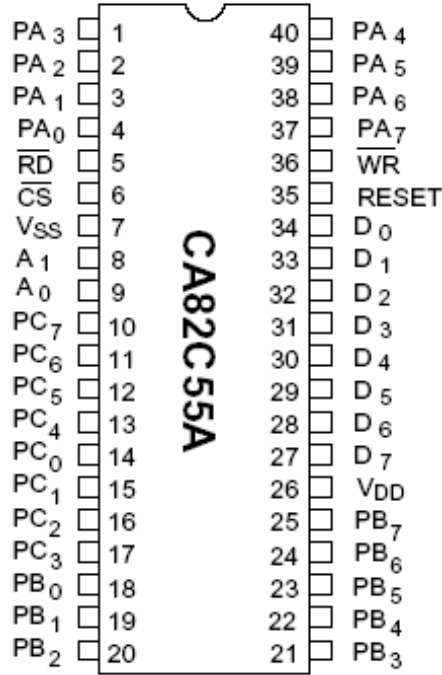
Birçok mikroişlemcili sistem, dış dünya ile haberleşmede basit giriş/çıkış (I/O) portları kullanmaktadır. Fakat çok sayıda porta ihtiyaç duyulması durumunda yetersiz kalmaktadır, ayrıca bu birimler sadece basit veri transferi için uygundur.

I-8255, Intel 8080 ailesi için geliştirilmiş, ancak genel mikrobilgisayar uygulamalarına da uygun, gelişmiş bir paralel iletişim arabirimidir. I-8255 içinde, dört tane port bulunmaktadır. A, B, Cüst ve Calt olarak adlandırılan bu portlardan, A ve B sekiz bitlidir. C ise dörder bitlik Cüst ve Calt portlarından oluşmaktadır [21].

I-8255'i kontrol etmek için 6 giriş bulunur (Şekil 2.9). Bunların görevleri şu şekildedir:

- Reset: Lojik 1'de etkindir. Reset girişinin 1 yapılması sonunda, A, B, C ve denetim kütüğünün içerikleri sıfır ile yüklenmiş olur.
- CS: Entegre seçim sinyalleridir. CS girişi lojik 0'da etkindir.
- RD: Okuma işlemini belirtir. Lojik 0'da etkindir.
- WR: Yazma işlemini belirtir. Lojik 0'da etkindir.

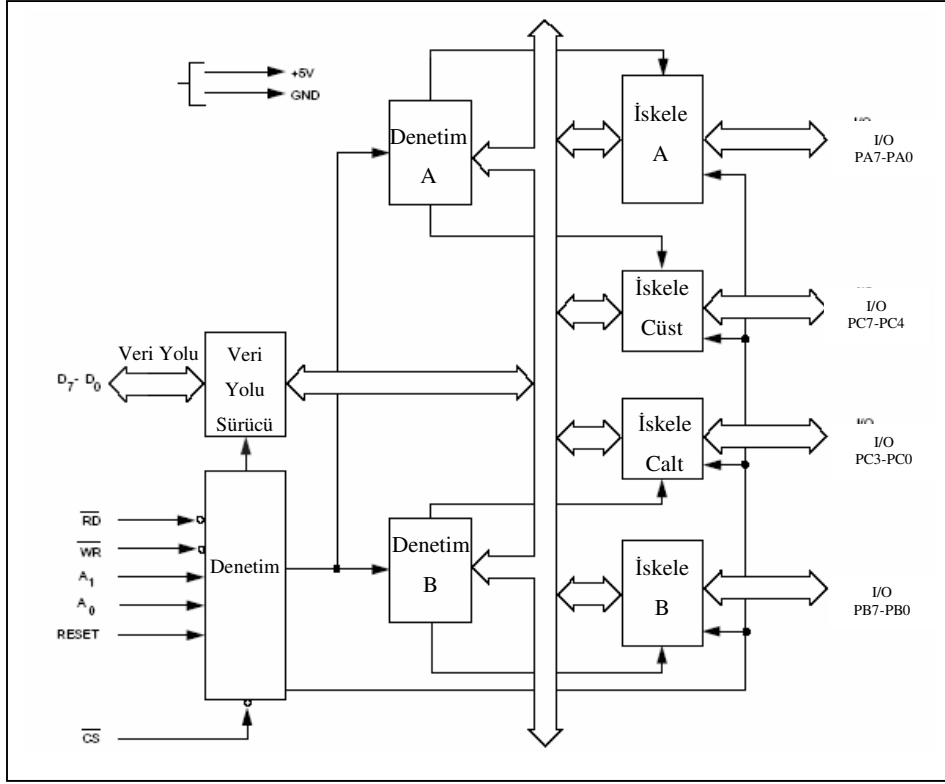
- A0-A1: A0 ve A1 girişleri I-8255 içindeki alt birimlerin seçilmesinde, okuma ve yazma girişleri ile birlikte kullanılır. A0, A1, RD, WR ve CS'nin alacağı değerlere göre seçimin nasıl yapıldığı Tablo 2.3'de gösterilmiştir. Aynı tabloda, seçim sonrasında, veri akış yönü de belirtilmektedir.



Şekil 2.9 8255 PIA Yongası

Tablo 2.3 I8255 içindeki alt birimlerin seçim yöntemi

A1	A0	RD	WR	CS	İŞLEM
0	0	0	1	0	A İskelesi → Veri Yolu
0	1	0	1	0	B İskelesi → Veri Yolu
1	0	0	1	0	C İskelesi → Veri Yolu
0	0	1	0	0	Veri Yolu → A İskelesi
0	1	1	0	0	Veri Yolu → B İskelesi
1	0	1	0	0	Veri Yolu → C İskelesi
1	1	1	0	0	Veri Yolu → Denetim Kütüğü



Şekil 2.10 8255 PIA iç yapısı

Veri Yolu: D0-D7 hatları ile MIB arasındaki iki yönlü iletişimi sağlar.

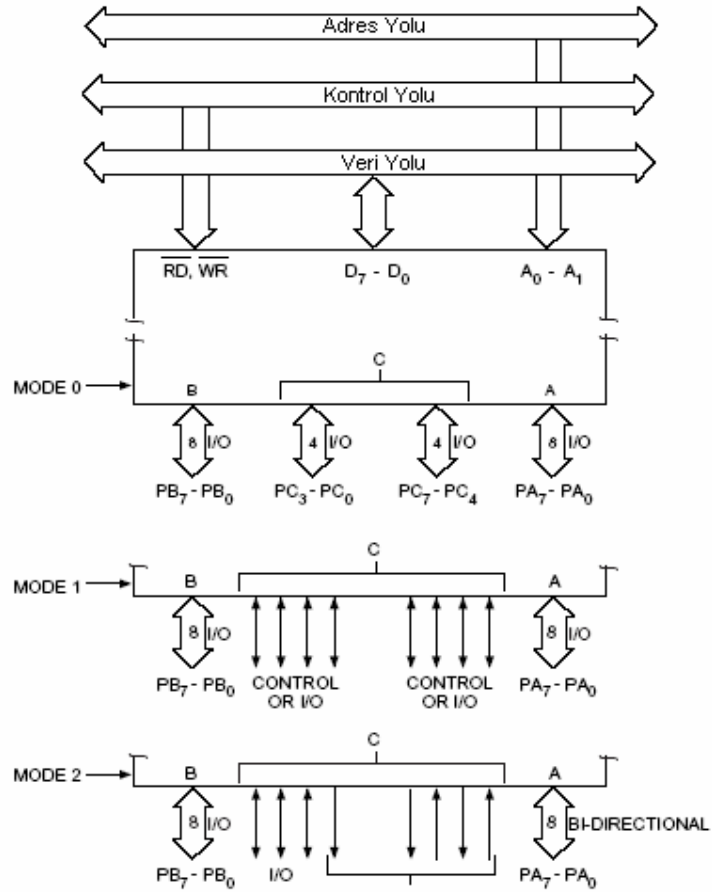
Portlar: Portların kullanım biçimleri, denetim kütüğü içine yazılan verilerle belirlenir.

I-8255'de fonksiyonlar iki moda göre sınıflandırılmıştır. Bit Set/Reset (BSR) modu ve I/O modu. Portların kullanım biçimleri, denetim kütüğü içine yazılan verilerle belirlenir. BSR modu C portundaki bitleri 1'lemek veya 0'lamak için kullanılır. I/O modu ise Mod 0, Mod 1 ve Mod 2 olarak üçe ayrılır. Bu modların nasıl sağlandığı Tablo 2.4'de gösterilmiştir.

Mod 0: A ve B portunun tüm kapıları ya alıcı ya da verici durumda konumlanır. C portunun kapıları ise, denetim kütüğünün D0, D1, D2 ve D3 bitlerine uygun olarak konumlanır.

Tablo 2.4 PIA'da denetim kütüğünün modları belirlemesi

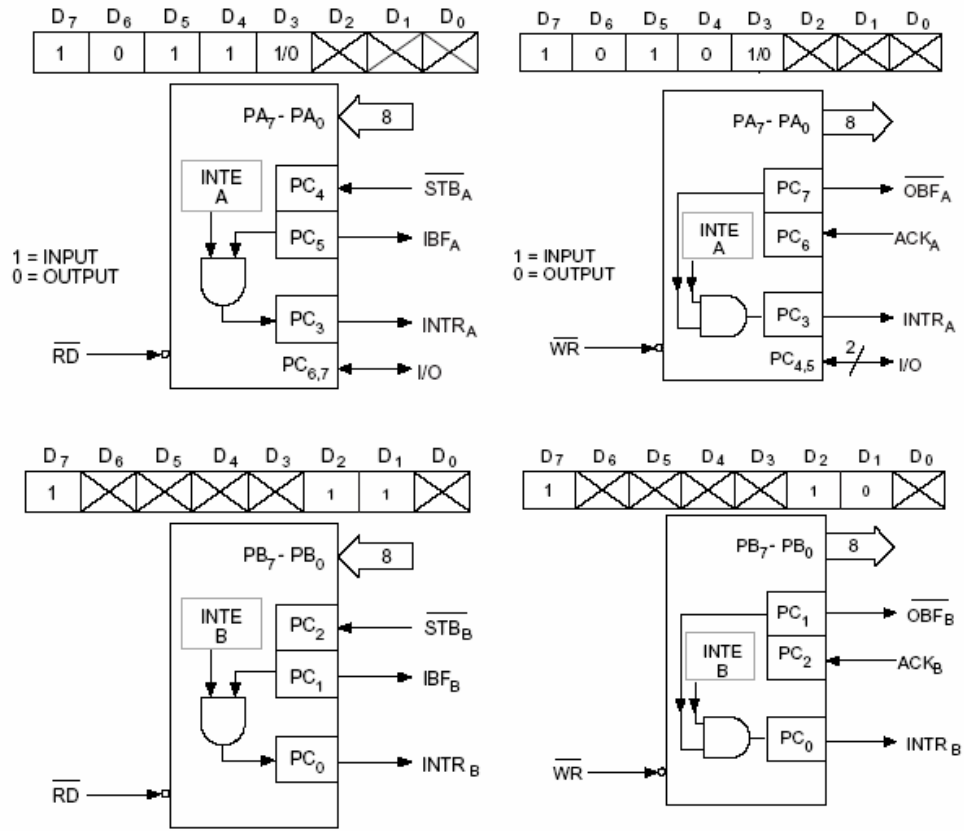
D0	Calt	1 Giriş 0 Çıkış
D1	B Portu	1 Giriş 0 Çıkış
D2	Mod Seçici	0 Mod 0 1 Mod 1
D3	Cüst	1 Giriş 0 Çıkış
D4	A Portu	1 Giriş 0 Çıkış
D5-D6	Mod Seçici	00 Mod 0 01 Mod 1 1X Mod 2
D7	Mod Kur Bayrağı	1 Etkin



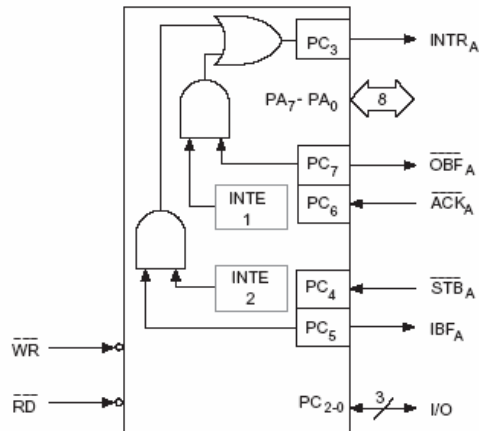
Şekil 2.11 Mod tanımları ve yol arayüzü

Mod 1: A ve B portlarının tüm kapıları ya alıcı ya da verici olarak konumlanır. C portunun üst kısmı A ve alt kısmı B portuna el sıkışma işlemleri için destek verir. PC2 B portu için hazır giriş olarak görev yapar. PC1 ise B portu için A1 çıkışı olarak görev yapar. Hazır bilgisinin alınması ile kesme üretilmek isteniyorsa PC0 bu amaçla

kullanılabilir. Mod 1 çalışma Şekil 2.12’de gösterilmiştir.



Şekil 2.12 Mod1 girişleri / çıkışları



Şekil 2.13 Mod2 girişleri / çıkışları



Mod 2: Sadece A portu ve Cüst için geçerlidir. Bu modda A portu iki yönlü olarak kullanılabilir (Şekil 2.13).

## 2.8. 8085 Mikroişlemci Komut Seti

8085A mikroişlemci komut seti, 74 farklı işlemi gerçekleştiren 246 komuttan oluşur. 8080A mikroişlemci komut setinde bulunan tüm komutlara sahip olan 8085A mikroişlemcisi, ek olarak iki seri giriş / çıkış komutuna (SIM ve RIM) sahiptir [22].

8085 mikroişlemcisinde komutlar, yapılan işlemler referans alınarak beş grup altında toplanabilir [22]:

1. Veri aktarım komutları
2. Aritmetik işlemler ile ilgili komutlar
3. Mantık işlemleri ile ilgili komutlar
4. Program akışı kontrol komutları
5. Mikroişlemci kontrol komutları

İşlevsel olarak gruplandırılmış 8085 mikroişlemcisi komutlarının adresleme türleri, kaç saykılta işlendikleri, etkiledikleri bayraklar ve kullanımları Ek-A'da verilmiştir.

Komutların uzunluklarına göre sınıflandırılması: 8085 mikroişlemci komut seti bellekte 1, 2 ve 3 bayt (8-bit) uzunlukta yer kaplayan komutlardan oluşur.

Tablo 2.5 Birbirinden farklı komut uzunluklarına sahip üç örnek komut

Komut Uzunluğu	İşkodu	İşlenen	Bellekte Yerleşimi	İkili Kodu	Anlamı
1 bayt	MOV	C, A	4F	0100 1111	A'yı C'ye kopyalar.
2 bayt	MVI	A,07H	3E 07	0011 1110 0000 0111	07 hex değeri A'ya yükler.
3 bayt	JMP	2085H	C3 85 20	1100 0011 1000 0101 0010 0000	1. Bayt 2. Bayt 3. Bayt

Bir bayt uzunluklu komutlar: İşkodunun ve işlenenin bir bayt uzunluğundaki bilgi ile tanımlandığı komutlardır. İki bayt uzunluklu komutlar: İlk bayt işkodunu, ikinci bayt ise işleneni temsil eder. Üç bayt uzunluklu komutlar: İlk bayt işkodunu, takip eden iki bayt ise 16 bit adresi veya veriyi tanımlar. İşlenen bilginin adres olması durumunda, ikinci bayt düşük değerli adresi, üçüncü bayt ise yüksek değerli adresi gösterir.

Intel 8085 mikroişlemcisinde gerçekleştirilen tüm işlemler, kaydediciler ve kaydedici çiftleri belirli kodlarla tanımlanmıştır. İşkodu, komut kodu ile birlikte işlenen kaydedici ya da kaydedici çiftlerinin kodlarından oluşur.

8085’de bulunan kaydediciler Tablo 2.6’da gösterilen kriterlere göre kodlanırlar.

Tablo 2.6 8085’de bulunan kaydedicilerin kodlanması

Kod	Kaydedici	Kod	Kaydedici çifti
000	B	00	BC
001	C	01	DE
010	D	10	HL
011	E	11	SP
100	H		
101	L		
111	A		
110	Bellekle ilgili işlemler		

Örnek olarak A kaydedicisinin içeriğini C kaydedicisine aktaran komut satırını inceleyelim:

Komut    Hedef Kaydedici    Kaynak Kaydedici  
 MOV    C                    A  
 01      001                    111  
 MOV C,A → 0100 1111 → 4F

Veri aktarma komutları: Bilginin bir kaydediciden diğerine veya bir bellek bölgesi ile

bir kaydedici arasında mümkün veri aktarımları, veri transferi komutları ile gerçekleştirilir. Verinin alındığı yer ‘kaynak’, verinin aktarıldığı / yazıldığı yer ise ‘hedef’ olarak tanımlanır. Bir komutun yazılmasında; işkodu kısmından sonra hedef ve kaynak kaydedicileri / adresleri yazılır. Hedef kaydedici / adres virgülden önce, kaynak kaydedicisi / adres ise virgülden sonra yazılır [22].

Bellekten kaydedicilerden birisine veri aktarımı işlemini yapan komutlar ‘yükleme komutları’ olarak adlandırılırken, mikroşlemci kaydedicilerinden belleğe veya giriş / çıkışa veri gönderme işlemini gerçekleştiren komutlar ‘saklama komutları’ olarak tanımlanır [22].

Aktarma işlemi kopyalama olarak düşünülebilir, çünkü aktarma işlemi kaynak kaydedicideki veya kaynak bellek bölgesindeki veriyi değiştirmez.

Aktarma komutları: Bu komutları aktarma, yükleme, yazma, giriş-çıkış ve takas komutları şeklinde alt gruplara ayırabiliriz.

Aritmetik komutlar: Aritmetik işlemleri gerçekleştiren komutlar ‘aritmetik işlem komutları’ olarak isimlendirilir. Aritmetik işlemler ile ilgili komutları; toplama, çıkarma, artırma / azaltma, ve döndürme komutları şeklinde alt gruplara ayırabiliriz.

Mantık komutları: Veriler üzerinde mantık işlemlerini (VE, VEYA, ÖZEL VEYA, karşılaştırma) gerçekleştirmek için kullanılırlar.

Program akışı kontrol komutları: Programın işlenmesindeki normal sırayı değiştirmek için kullanılır.

Mikroişlemci kontrol komutları: NOP komutu sadece program sayıcıyı artırır ve genellikle programlarda zaman gecikmesi sağlamak amacıyla kullanılır.

## **BÖLÜM 3. PIC16F84 MİKRODENETLEYİCİLER**

### **3.1. Mikrodenetleyici Kavramı**

Mikroişlemciyle (CPU) birlikte kullanılan giriş / çıkış ve bellek birimlerinin bir arada kullanılmasını sağlayacak şekilde bir yonga içine yerleştirilmesi ile oluşan eleman, ‘mikrodenetleyici’ (microcontroller - MCU) olarak isimlendirilir [22].

Çevresel Üniteleri Denetleyici Arabirimi olarak tercüme edilen PIC (Peripheral Interface Controller – PIC) mikrodenetleyicisi gerçekten de çevresel üniteler adı verilen lamba, motor, röle, ısı ve ışık sensörü gibi I/O elemanlarının denetimini çok hızlı olarak yapabilecek şekilde tasarlanmış bir entegredir [19].

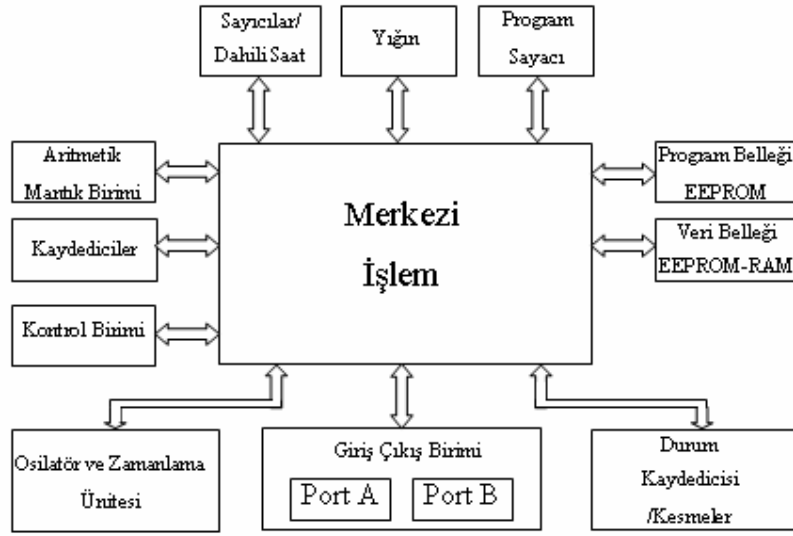
Mikroişlemci ile mikrodenetleyicileri birbirinden ayıran en önemli fark; mikroişlemci ile birlikte kullanılan program ve veri bellekleri ile giriş / çıkış birimlerinin tek bir entegre içerisine yerleştirilmesidir [22].

#### **3.1.1. Mikrodenetleyicilerin genel özellikleri**

Günümüz mikrodenetleyicileri genel olarak aşağıdaki birimlerden oluşur [22]:

- Merkezi işlem birimi – MİB (Central Processing Unit-CPU)
- Program belleği (RAM)
- Veri belleği (RAM ve ROM)
- Paralel Giriş / Çıkış terminalleri / portları
- Analog – Dijital çevirici (ADC)
- Dijital – Analog çevirici (DAC)
- Zamanlayıcılar ve sayıcılar

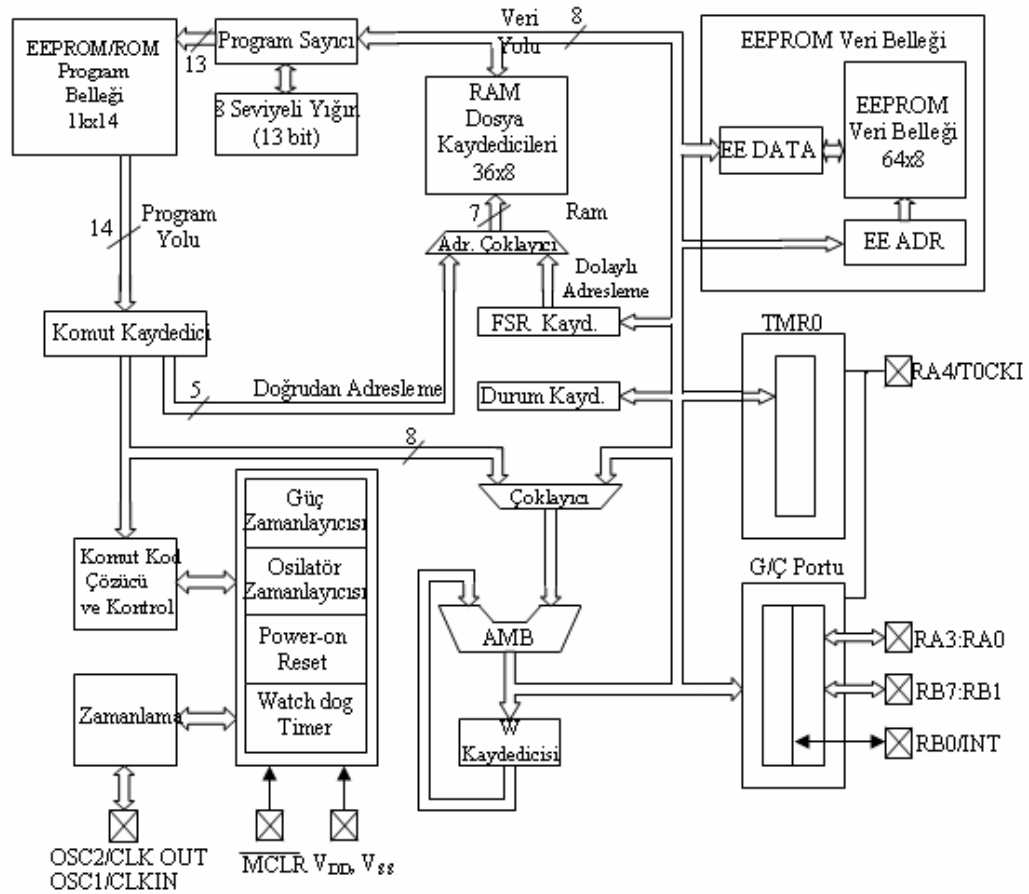
- Universal asenkron verici – alıcı (UART)
- Seri haberleşme birimi (RS-232, CAN, I2C, vb.)
- Pals genişlik üretici (PWM)
- Kesme kontrol birimi
- Doğrudan Bellek Erişim birimi (DMA)
- Güç yönetim birimi
- Diğer çevresel birimler



Şekil 3.1 Mikrodenetleyici genel blok diyagramı (Ekiz 2005)

PIC16F8X, PIC16CXX mikrodenetleyici ailesinin ucuz, yüksek performanslı, CMOS, tamamen statik, 8-bitlik mikrodenetleyici grubudur. Bu gruptaki mikrodenetleyiciler PIC16F83, PIC16F84, PIC16CR83 ve PIC16CR84'tür.

Tüm mikrodenetleyiciler geliştirilmiş RISC mimarisinde üretilmiştir. PIC16F8X cihazlarında artırılmış özelliklerine sahiptir. 8 seviyeli yığın artırılmış dahili ve harici kesme kaynakları. Harvard mimarisine göre 14 bitlik komut kelimesi ile 8 bitlik veri yolları birbirinden ayrılmıştır. Toplam 35 komut vardır. Buna ilave olarak geniş kullanıcı kaydedici alanı (RAM) yüksek performans elde edilmesini sağlamıştır. PIC16F84 mikrodenetleyicisi 68 bayt RAM, 64 bayt Data EEPROM belleği, 13 I/O ayağı ve bir adet zamanlayıcı/sayıcı'ya sahiptir [23].



Şekil 3.2 PIC16F8X Blok Diyagramı (Microchip 2001)

### 3.1.2. Neden PIC mikrodenetleyiciler

PIC mikrodenetleyicilerin tercih edilmesinin ve hızlı bir gelişim göstermesinin nedenleri / etkenleri olarak aşağıdaki hususlar sıralanabilir [22]:

- Gerekli yazılımın ve simülasyon programların ücretsiz olarak sağlanması.
- Yaygın kullanıma sahip olması nedeni ile PIC'i bilen çok sayıda insanın bulunması.
- Ucuz ve kolay olarak bulunabilmesi.
- Bellek bölgelerine erişimde (adresleri iletmek amacıyla) ve veri iletiminde farklı / ayrı yolların kullanılması.
- Basit elemanların eklenmesiyle oluşturulan sistem / donanım yardımıyla programlanabilmesi.

- Kullanıldığı devrelerde basit yapıda yardımcı devrelere (sıfırlama, tetikleme / saat sinyali, besleme, vb) ihtiyaç duyulması.
- Yüksek frekanslarda çalışabilmesi ve komut işleme hızının çok yüksek olması.
- RISC komut mimarisi kullanılması nedeni ile az sayıda ve basit yapıda komutlara sahip olması (PIC 16F84’de 35 komut kullanılmaktadır).
- ‘Stand-by’ durumunda çok az güç harcaması (yaklaşık 1mA akım çekmesi).
- Dahili ve harici osilatör ile çalıştırılabilmesi.
- ‘Harward’ mimarisine sahip olması nedeniyle komutlar ve verinin farklı yollar kullanılarak iletilmesi ve işlemleri hızlı olarak gerçekleştirebilmesi.
- Sistem oluşturmak için çok az sayıda elemana ihtiyaç duyulması (sadece 2 kondansatör ve 1 adet direnç yeterli olabilir).
- Kesme özelliğinin bulunması.
- Komut işleme belleğine (14 bit) sahip olması.
- G / Ç uçlarından elektronik elemanları rahatlıkla sürebilmesi.

### 3.1.3. Neden PIC16F84

En önemli nedeni: PIC16F84 mikrodenetleyicisinin program belleğinin Flash teknolojisi ile üretilmiş olmasıdır.

EEPROM – Elektriksel olarak silinebilen ve programlanabilen bellek (Electrically Erasable Programmable Read Only Memory): Çoğu mikrodenetleyici ailesinde sınırlı sayıda EEPROM tipi bulunur. EEPROM özelliğine sahip olanlar, birden fazla yazılıp silinebildiği için gayet kullanışlıdır. Daha çok zamanla programında değişiklik gerekebilecek sistemlerde kullanılırlar. Tekrardan yazma ve silme ömürleri sınırlıdır.

FLASH EPROM - Flash EPROM içeren bellekler, EEPROM’lardan daha hızlı ve daha çok yazma silme işlemine izin vermeleri yönünden üstündürler.

OTP – Bir kez programlanabilir (One Time Programmable): OTP bir kez programlanabilen bir ROM’dur. Programınızı bir EPROM programlayıcı ile bir kez yazdıktan sonra silemez ya da değiştiremezsiniz. Bu yöntem programınız artık

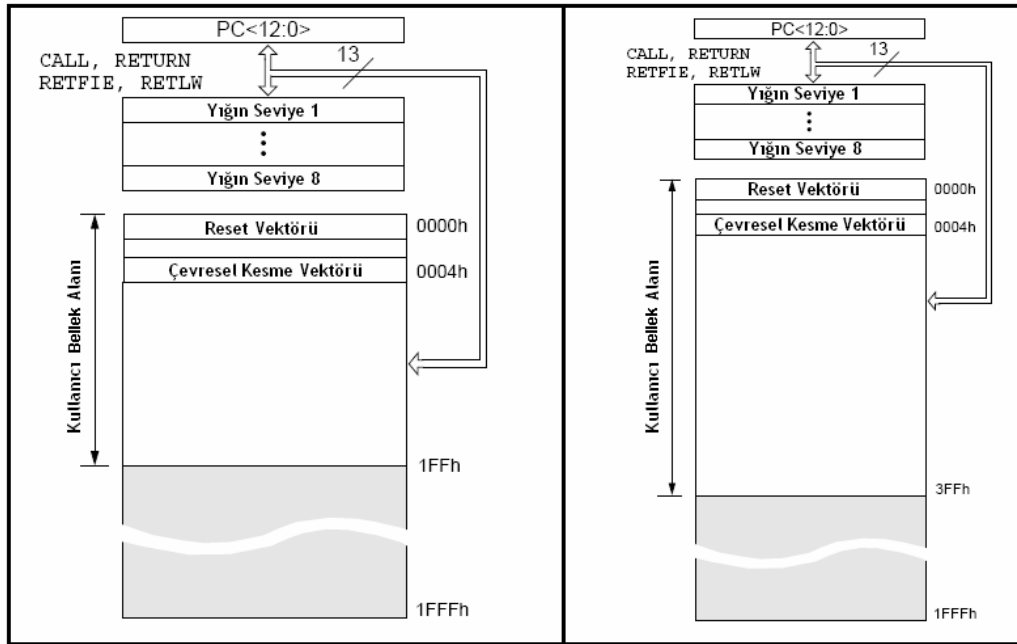
tamamen hazır olduğunda ve bütün hatalarından arındırıldıktan sonra kullanılır. Seri olarak çok sayıda üretilen mikrodenetleyicili sistemlerde, maliyeti düşürmek için, bu düşük maliyetli OTP cihazlar kullanılır [17].

PIC16F84 Kullanım Alanları: Yüksek hızlı otomasyon, düşük güçlü kumanda sensörleri ile motor kontrol, elektronik kilitler, güvenlik cihazları ve smart kartlar. Flash / EEPROM teknolojisi programlarının özelleşmesini sağlamıştır.

### 3.2. Bellek Organizasyonu

#### 3.2.1. Bellek türleri

PIC16F8X’de iki bellek bloğu vardır; program belleği ve veri belleği. Her bir blok kendi veri yolu sistemine sahiptir. Bundan dolayı her bir bloğa aynı osilatör çevriminde ulaşım mümkündür.



Şekil 3.3 Program Bellek Haritası ve Yığın (Microchip 2001)

Veri belleği “Genel Amaçlı RAM” ve “Özel Fonksiyon Kaydedicileri” (Special Function Register-SFR) olmak üzere iki kısımda incelenebilir. SFR’lerin işlemleri bünyesinde kontrolün çekirdeğini barındırmaktadır. SFR’ler çevre birimlerini kontrol



ederler.

Veri belleđi alanı ayrıca EEPROM belleđini de ierir. Bu belleđe direk olarak ulařılmaz, dolaylı ulařım söz konusudur. Bunun iin bir dolaylı adres gsterici veri EEPROM belleđine okuma/yazma iřlemi iin zelleřtirilmiřtir. 64 baytlık veri EEPROM belleđi 0h-3Fh adres aralıđına sahiptir.

### 3.2.2. Program belleđinin organizasyonu

PIC16FXX 13 bit program sayacına sahiptir ve 8Kx14 boyutunda bir bellek alanını adresleyebilmeyi sađlar. 16F84 ve 16CR84'te ilk 1Kx14 (0000h-03FFh)'lık kısım fiziksel olarak mevcuttur. Fiziksel olarak var olmayan bir adrese yazmaya alıřılırsa fiziksel olarak var olan bir adrese, ieriđine bakılmadan yazma iřlemi gerekleřir. rneđin 16F84'te 20h adresi, 420h, 820h, C20h, 1020h, 1420h, 1820h ve 1C20h adreslerini etkileyecek komuttan etkilenebilir.

Bunun yanında reset vektr 0000h adresinde, interrupt (kesme) vektr ise 0004h adresindedir.

### 3.2.3. Veri belleđi organizasyonu

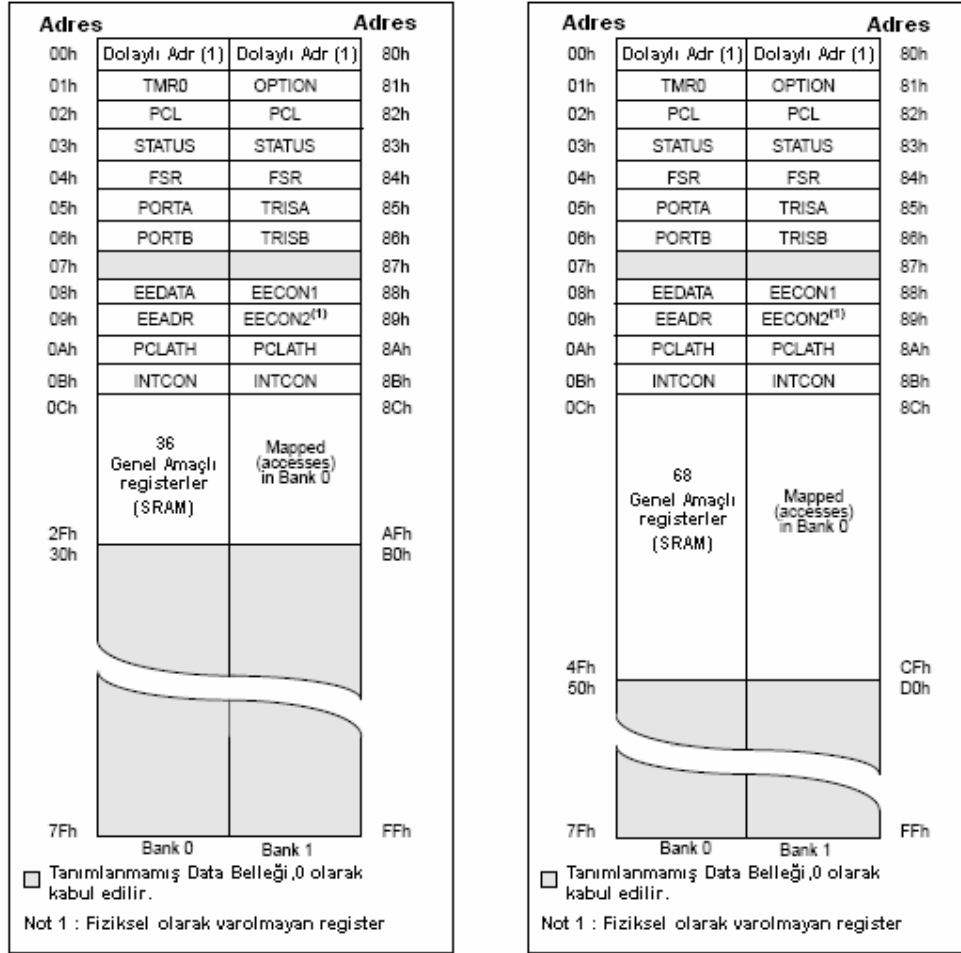
Veri belleđi iki alana blnmřtr. İlki zel Fonksiyon Kaydedici (SFR) alanı, ikincisi ise Genel Amalı Kaydedici (GPR) alanıdır. SFR'ler cihazların iřlemlerini kontrol eder.

Veri belleđinin bu iki kısmı da belli bir adres aralıđı ile sınırlandırılmıřtır. Bu iki kısma ulařmak iin kontrol bitleri kullanılır. Bu kontrol bitleri STATUS kaydedicisinde bulunmaktadır. Őekil 3.4'de denetleyicilere gre veri belleđinin haritası gsterilmektedir.

Genel Amalı Kaydedici Dosyası: Tm cihazlar Genel Amalı Kaydedici (GPR) zerinde tanımlanırlar. Her bir GPR 8 bit geniřliđindedir ve dođrudan veya FSR ile dolaylı olarak adreslenebilirler. GPR'nin bank1 ve bank0 adresleri simetriktir. rneđin 0Ch veya 8Ch adresi aynı GPR blgesine eriřecektir.

Özel Fonksiyon Kaydedicileri: Özel Fonksiyon Kaydedicileri CPU ve çevre birimleri tarafından cihaz işlemlerini kontrol amaçlı kullanılır. Bu kaydedici statik RAM'dir. SFR iki grupta sınıflandırılabilir; çekirdek ve çevre birimler.

PIC16F84 mikrodenetleyicisinin status, option, intcon ve eecon özel kaydedicileri bit düzeyinde işlevleri ve açıklamaları ile Ek-B'de ayrıntılı olarak verilmiştir.



Şekil 3.4 Kaydedici Dosya Haritası (Microchip 2001)

### 3.2.4. Özel fonksiyon kaydedicileri

STATUS kaydedicisi: ALU (Aritmetik Lojik Ünitesi)'nun aritmetik durumu, RESET durumu ve veri belleği için bank seçme bilgisini içerir.

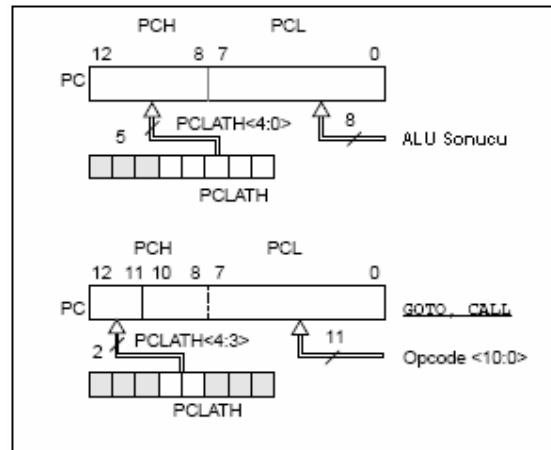
Diğer kaydedicileri gibi STATUS kaydedicisi herhangi bir komut için hedef olabilir. Z, DC veya C bitleri etkilendiğinden bu üç bite yazma işlemi gerçekleştirilemez. Bu üç bit cihaz mantığı ile set veya temizleme işlemi olur. Bunun yanında PD ve TO bitleri de üzerine yazma işlemi kabul etmez. Bundan dolayı bir komut ile kaydediciye bir şey yazma istendiğinde STATUS kaydedicisinde olması tasarlanan bilgi görülemeyebilir.

Örneğin, “CLRf STATUS” komutu yüksek üç biti temizler, Z bitini set eder ve STATUS kaydedici içeriği “000u u1uu” (u-unchanged değişmez) şeklinde olur.

OPTION kaydedici: Üzerine hem yazma işlemi hem de kaydedici okuma işlemi kabul eder. Bu kaydedici TMR0/WDT prescaler, harici INT kesmesi, TMR0 ve PORTB'nin pull-up dirençlerinin konfigürasyon bitlerini içerir.

INTCON kaydedici: Okunabilir ve yazılabilir bir kayıt dosyasıdır ve bünyesinde tüm kesmelerin yetkilendirme bitlerini içerir.

Program Sayacı: 13 bit genişliğindedir. PCL kaydedicinin düşük baytı okunabilir ve yazılabilir bir kaydedicidir (Şekil 3.5).



Şekil 3.5 Farklı durumlarda PC'nin yüklenmesi (Microchip 2001)

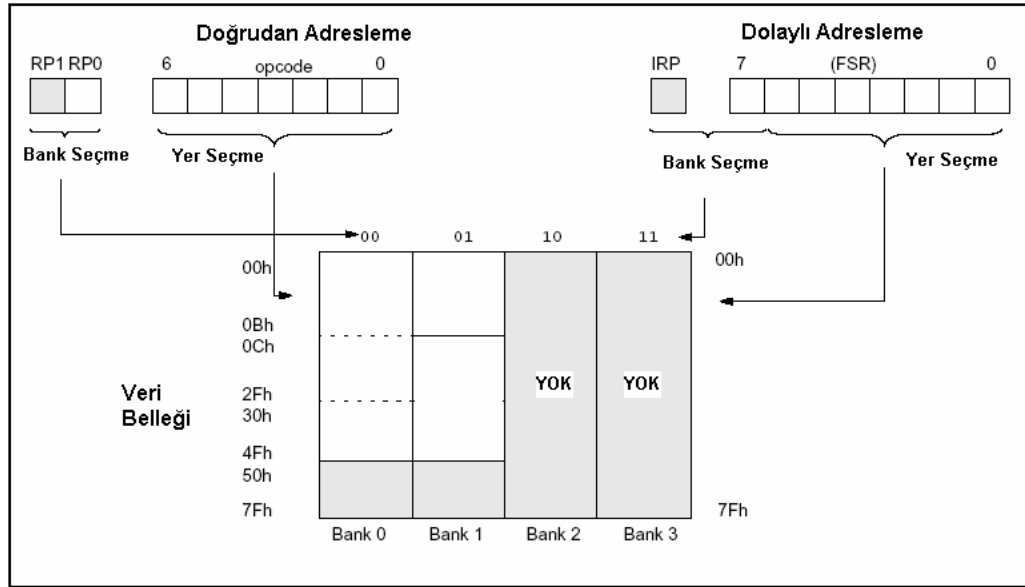
PC'nin yüksek bitleri (PC<12:8>) doğrudan ne okunabilir ne de yazılabilir. Bu kaydedici PCLATH kaydedici olarak adlandırılır. PCLATH kaydedici (PC LATH

yüksek) PC'nin yüksek bitlerini tutar. PC'ya yeni bir değer yüklendiği zaman yüksek bitler PCLATH'a aktarılır.

Dolaylı adresleme INDF ve FSR kaydedicileri: INDF kaydedicisi fiziksel olarak mevcut değildir ve bunun adresleme işlemi FSR ile yapılır (FSR bir adres göstericidir). Buna dolaylı adresleme denir.

Aşağıda örnek bir dolaylı adresleme verilmiştir:

- 05 adresi 10h değerini içersin
- 06 adresi 0Ah değerini içersin
- 05 adresini FSR kaydedicisine yükleyelim
- INDF kaydedicisi okunduğunda 10h değeri görülür
- FSR kaydedicisini 1 artıralım (FSR 06 olur)
- INDF kaydedicisi şimdi okunduğunda 0Ah değeri görülür.



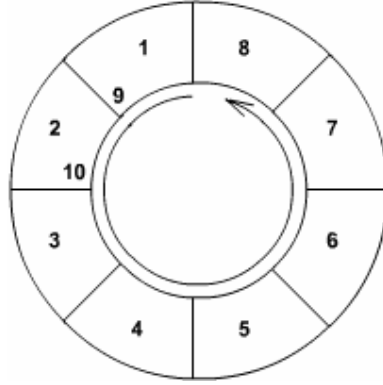
Şekil 3.6 Doğrudan ve Dolaylı Adresleme (Microchip 2001)

### 3.2.5. Yığın

PIC16FXX, 13 bit genişliğinde 8 seviyeden oluşan bir yığına sahiptir. Yığın, ne program ne de veri yüzeyinin bir parçası değildir ve yığın gösterici ne okunabilir ne de yazılabilir. 13 bitlik Program Sayacının içeriğinin tamamı, CALL komutu

işlendiğinde veya bir kesme kabul edildiğinde yığına atılır. RETURN, RETLW veya RETFIE komutları işlendiğinde ise yığın içeriği geri alınır.

Yığını dairesel tampon olarak görebiliriz. Şöyle ki yığının 8 seviyesini de kullanmış olalım ve yığına 9. bir bilgi atılmış olsun. Bu bilgi yığının ilk bölümüne yazılır. 10. ise ikinci bölümün üzerine yazılır. Bu döngü böylece devam eder.



Şekil 3.7 Yığının dairesel tampon gösterimi

### 3.3. Giriş / Çıkış Portları

Mikrodenetleyici, sistemde bulunan diğer elemanları veya cihazları izlemek veya kontrol etmek amacıyla portları kullanır. PIC16F84 mikrodenetleyicisinde standart olarak PORTA ve PORTB olmak üzere iki adet giriş/çıkış portu bulunmaktadır. 18 pine sahip olan PIC16F84 mikrodenetleyici entegresinde RA0-RA4 olarak tanımlanan 5 tanesi PortA ve RB0-RB7 sembolleri ile gösterilen 8 tanesi PortB olarak kullanılan toplam 13 adet giriş/çıkış pini bulunmaktadır.

PIC16F84 entegresinin 3 no'lu ucuna karşılık gelen A portunun 4. biti, 'TOCKI' olarak isimlendirilen harici zamanlayıcı/sayıcı giriş ucu ile zaman paylaşımli olarak kullanılır. Bu nedenle PIC16F84 entegresinin 3 no'lu pini üzerinde RA4/TOCKI yazılır. Harici kesme işleminin aktif yapılması durumunda 6 no'lu pin kesme işlemi için kullanılır.

PortA yalnızca 5 pine sahiptir ve PortA'nın 5 nolu pini RA4 yalnızca giriş olarak

kullanılabilir. RA4 pini ikinci işlevine bağlı olarak TMR0 zamanlayıcısı için harici giriş olarak ta kullanılabilir. RA4 pininin standart girişi yoksa sayıcı için tetikleme girişimi olarak kullanılacağı TOSC bitine (TMR0 saat kaynağı seçme biti) bağlıdır. TOSC biti, RA4/TOCKI pinindeki harici tetikleme girişi üzerinden veya dahili osilatörden aldığı palsler ile ‘TMR0’ zamanlayıcısını çalışması için uygun duruma getirir.

PortB’de bulunan RB4-RB7 arasındaki dört pinin giriş olarak belirlendiği durumlarda, pinlerin değerlerinin lojik 1’den lojik 0’a veya lojik 0’dan lojik 1’e değişmesi ile ‘kesme’ durumları oluşur.

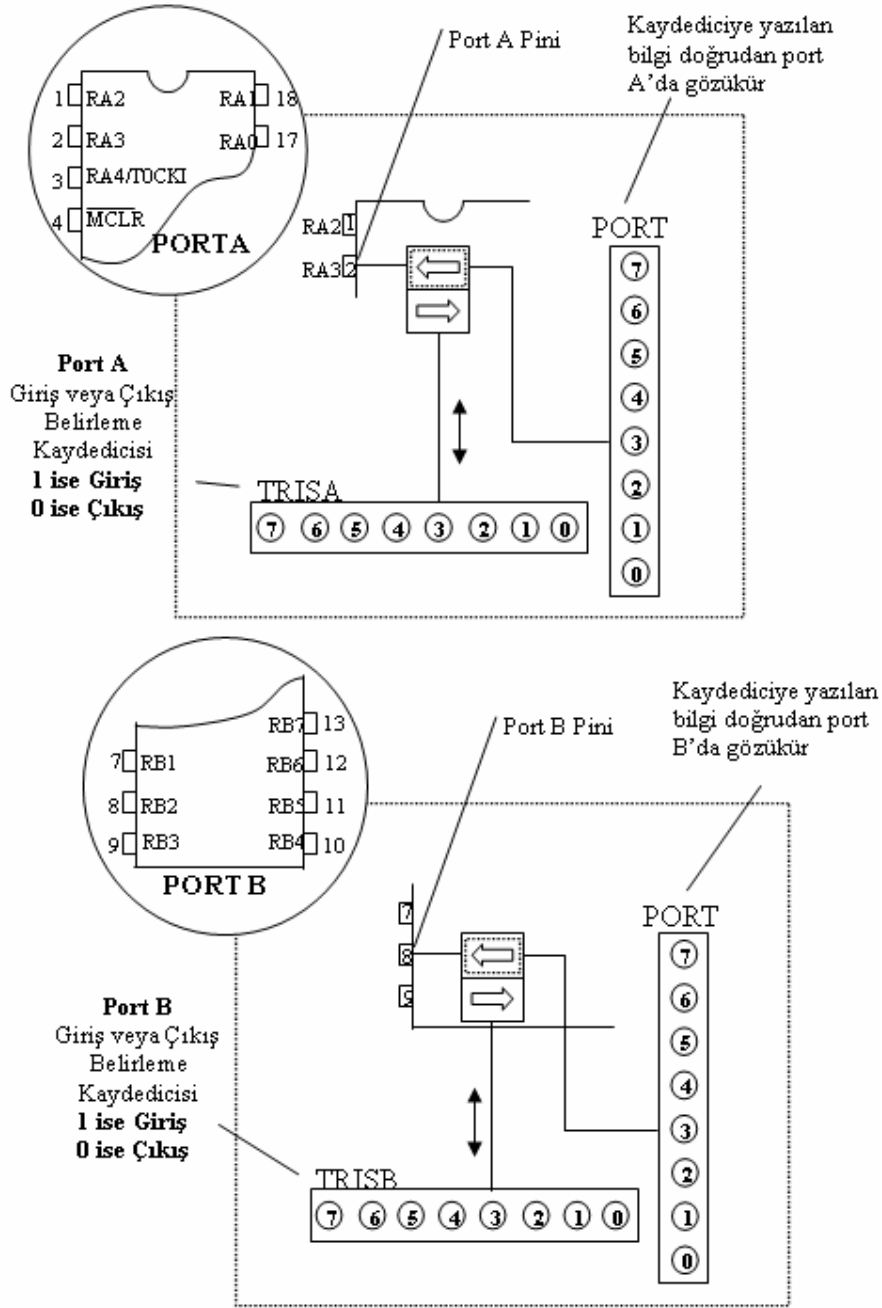
PortB’deki pinlerde normal durumda bulunan ‘Pull-Up’ dirençleri (hattın ‘lojik 1’ olmasını sağlar) ‘seçenek – option’ kaydedicisinin yedinci biti RBPU’nun ‘lojik 0’ yapılması ile aktif hale getirilir. PortB’de bulunan pinlerin çıkış olarak atanması durumunda ‘Pull-Up’ dirençleri otomatik olarak devre dışı bırakılması yanında mikrodenetleyicinin çalışmaya başladığı durumlarda ‘Pull-Up’ dirençleri ‘yetkisiz’ hale getirilir.

PortA ile ilgili TRISA ve PortB ile ilgili TRISB kaydedicilerinde bir bitin ‘lojik 1’ yapılması ilgili porttaki bitin giriş olarak ve bitin ‘lojik 0’ yapılması ilgili bitin çıkış olarak belirlenmesini sağlar (Tablo 3.1) [22].

Tablo 3.1 PORTA – PORTB Fonksiyonları

Adı	Fonksiyonu
RA0-RA3	Giriş/çıkış
RA4/TOCKI	Giriş/çıkış -TMR0 harici saat girişi. Çıkış open-drain tipi
RB0/INT	Giriş/çıkış - harici kesme kaynağı.
RB1-RB3	Giriş/çıkış, yazılımla pull up yetkilendirilebilir.
RB4-RB7	Giriş/çıkış ve pinin değişmesi ile kesme kaynağı, pull up

Her bir port ile ilgili pinlerin yönlerinin belirlenmesi amacıyla kullanılan bir adet TRIS kaydedicisi bulunmaktadır: PortA ile ilgili TRISA ve PortB ile ilgili TRISB (Şekil 3.8).



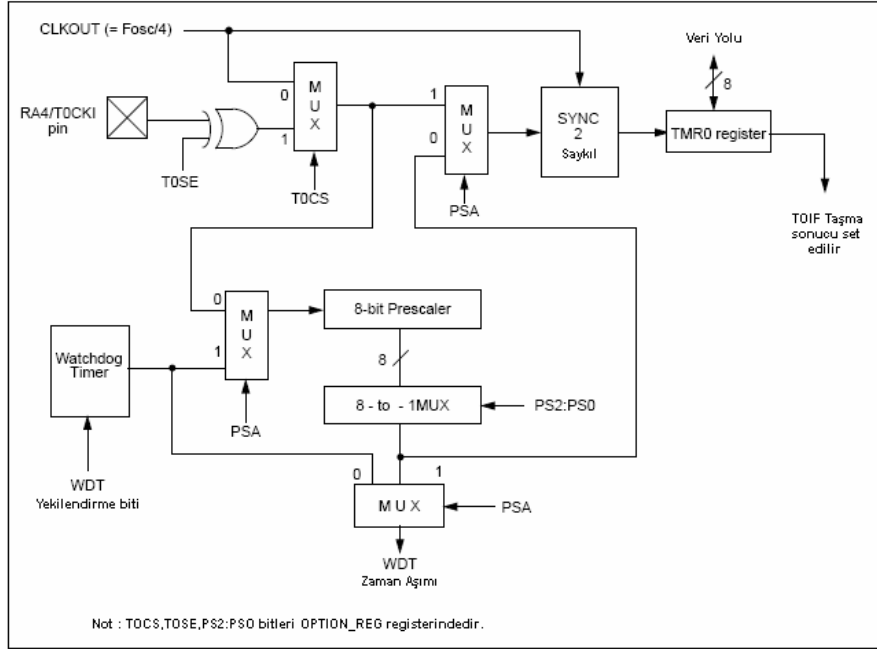
Şekil 3.8 TRIS kaydedicilerinin yapısı ve bitlerin işlevleri (Ekiz 2005)

### 3.4. TIMER0 Modülü ve TMR0 Kaydedicisi

#### 3.4.1. Timer0 modülü

Timer0 modülü zamanlayıcı/sayıcı olarak Şekil 3.9'da gösterilen özelliklere sahiptir;

- 8 bit zamanlayıcı/sayıcı
- Okunabilir ve yazılabilir özelliği
- Dahili ve harici saat seçimi
- FFh'tan 00h'a taşma kesmesi özelliği
- Harici saat için yükselen kenar veya düşen kenar pals seçimi



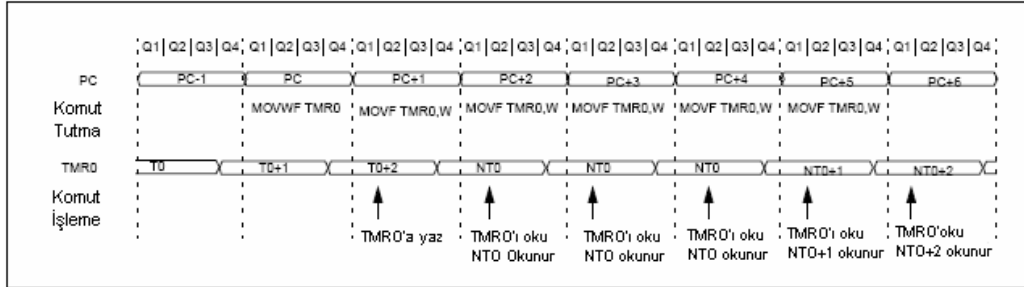
Şekil 3.9 TMR0 / WDT Prescaler Blok Diyagramı (Microchip 2001)

Timer modu OPTION\_REG kaydedicisinin TOCS bitini temizlemek suretiyle seçilmiş olur. Timer modunda, Timer0 modülü prescaler yok ise her bir komut çevriminde arttırılır (Şekil 3.10). Şayet TMR0 kaydedicisine yazma işlemi gerçekleşirse Şekil 3.10 ve Şekil 3.11'de gösterildiği gibi iki çevrim boyunca artma işlemi engellenir. Kullanıcı bu süre zarfında TMR0 kaydedicisine değer atayarak ayarlama yapabilir.

OPTION\_REG<5> kaydedicisinin TOCS biti set edilirse sayıcı modu seçilmiş olur. Bu modda RA4/TOCKI pinine uygulanan sinyalin ya yükselen ya da düşen kenar durumunda TMR0 da değer artışı gerçekleşir. Bu kenar seçimi aynı şekilde OPTION\_REG<4> kaydedicinde TOSE biti sayesinde gerçekleşir. TOSE=0 ise yükselen kenar, TOSE=1 ise düşen kenarda aktivite gerçekleşir.

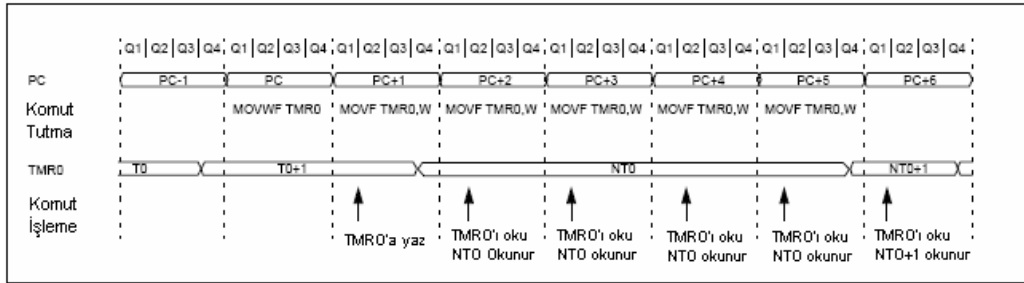


TMR0 için harici saat kullanılacaksa mutlaka gerekli olan şeyler sağlanmalıdır. Şöyle ki; harici saat dahili saat tarafından oluşturulan  $T_{OSC}$  ile eş zamanlı olmalıdır. Gerçekte TMR0 kaydedicisinin içeriğini arttırması bu senkronizasyonun belli bir gecikmesinden sonra olmaktadır.



Şekil 3.10 TMR0 Zamanlaması: Dahili Saat / Prescalersiz (Microchip 2001)

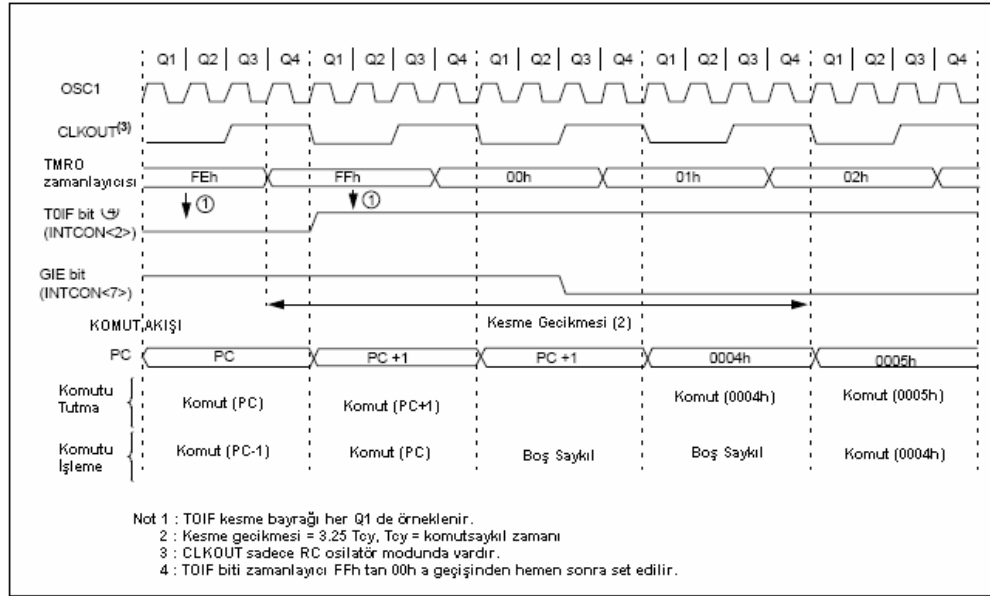
Prescaler, TMR0 modülü ve Watchdog Timer arasında paylaşılr. `OPTION_REG<3>` kaydedicisinin PSA bitini yazılımla kontrol ederek gerçekleştir.  $PSA=0$  ise prescaler Timer0 modülü için ayrılmış olur. Prescaler ne okunabilir ne de yazılabilir. Prescaler Timer0 modülüne tahsis edildiğinde prescaler değeri yazılımla seçilebilir.



Şekil 3.11 TMR0 Zamanlaması: Dahili Saat / Prescaler 1:2 (Microchip 2001)

### 3.4.2. TMR0 kesmesi

TMR0 kaydedicisinin değeri FFh'dan 00h'a bir taşma gerçekleştiğinde TMR0 kesmesi oluşur. Bu taşma `INTCON<2>` kaydedicisinde TOIF bitini set eder. Kesme maskelenmek isteniyorsa `INTCON<5>` kaydedicisinde TOIE biti temizlenmelidir. Bir kesmeden sonra sistemin tekrar kesmeye açık olması için TOIF bit mutlaka temizlenmelidir. TMR0 kesmesi SLEEP komutu boyunca zamanlayıcı kapatılmış ise işlemci uyanamaz.



Şekil 3.12 TMR0 Kesme Zamanlaması (Microchip 2001)

### 3.5. DATA EEPROM

Data EEPROM Belleği: Okunabilir ve yazılabilir bir bellektir. Ancak doğrudan adresleme yapılamamaktadır. Özel Fonksiyon Kaydedicileri kullanılarak dolaylı adresleme ile bu belleğe ulaşılır. Bu belleğe okuma/yazma işlemlerinde EECON1, EECON2, EEDATA ve EEADR olmak üzere 4 Özel Fonksiyon Kaydedicisi kullanılmaktadır.

EEDATA, okuma veya yazmada 8 bitlik veriyi, EEADR ise EEPROM'un hangi bölgesine ulaşılacak isteniyorsa o adresi tutar. PIC16F8X serisi 00h ile 0Fh aralığında 64 bayt EEPROM belleğine sahiptir.

Eğer işlemci kod korumalı olarak programlanmış ise CPU, EEPROM belleğine okuma ve yazma işlemi için erişim sağlayabilir ancak programcı ulaşamaz.

EEADR kaydedicisi: EEPROM'un maksimum 256 baytını adresleyebilecek kapasitededir. Sadece 64 baytlık EEPROM geliştirilmiştir. Yüksek iki bit adres kod çözücüdür ve bu bitler daima "0" olmalıdır ki EEPROM'un 64 bayt olarak adreslenir.

EECON1 kaydedicisi: Fiziksel olarak düşük değerlikli 5 bite sahiptir. Yüksek değerlikli 3 bit fiziksel olarak yoktur ve 0 olarak okunur.

Kontrol bitleri olan RD ve WR, okuma ve yazma işlemini gerçekleştirirler ve bu bitler yazılım ile temizlenemezler ancak set edilebilirler. Bu bitler okuma ve yazma işlemi tamamlandığında donanım tarafından temizlenir. WR bitinin yazılım tarafından temizlenemez olması yazma işleminin zamanından önce sonlandırılmasını engellemesi gibi faydası vardır.

WREN biti set edildiğinde yazma işlemi başlayabilir. Power-On durumunda WREN biti temizlenir. WRERR biti ise normal işlem sırasında MCLR reseti veya WDT zaman aşımı reseti ile program resetlenirse yazma işlemi kesilir ve bu bit set edilir. Böyle durumlarda programcı WRERR bitini kontrol etmeli ve aynı yazma işlemini tekrar yapmalıdır. EEDATA ve EEADR kaydedicileri bu durumdan etkilenmez ve kayıtlarını aynı şekilde tutarlar. Yazma işlemi tamamlandığında kesme bayrağı EEIF set edilir ve bu bayrak yazılım ile temizlenmelidir.

EECON2 fiziksel olarak gerçekten var olmayan bir kaydedicidir. 0 olarak okunur. EECON' veri EEPROM'a yazma işleminin rutin sırasında kullanılır.

EEPROM Okuma İşlemi: Bir hafıza bölgesini okumak için öncelikle EEADR kaydedicisine adres yazılır ve okuma kontrol biti RD set edilir (EECON1<0>). Son adım olarak EEDATA kaydedicindeki bilgi EEPROM'dan alınıp işlenebilir.

### 3.6. PIC16F84 Komut Seti

PIC 16F84 mikrodenetleyicisi komut setinde 35 adet komut bulunmaktadır. RISC (azaltılmış komut seti) kullanılması nedeniyle komutlarının öğrenilmesinin ve program yazılmasının kolay olması yanında programın kısa sürede yazılabilmesini sağlar. Komutların büyük bir kısmı 1 saat çevrimi sırasında gerçekleştirilirken, test ve dallanma komutları 2 saat çevrimi sırasında gerçekleştirilir.

Komutların işlevleri referans alınarak yapılabilecek gruplandırma/sınıflandırma şu

şekildedir [22] :

- Veri aktarımı / transferi işlemleri ile ilgili komutlar
- Aritmetik ve mantık işlemler ile ilgili komutlar
- Program akışı kontrol komutları
- Bit işlemleri ile ilgili komutlar
- Özel işlemlerle ilgili komutlar

Ayrıca W kaydedicisinin işlevinin hatırlanması, komutların çoğunda işlem yapabilmek için W kaydedicisine ihtiyaç duyulması nedeni ile önemli bir yere sahip olacaktır.

Komutların yazılış biçimini dört grupta toplayabiliriz [19].

1. Byte-yönlendirmeli komutlar
2. Bit-yönlendirmeli komutlar
3. Sabit işleyen komutlar
4. Kontrol komutları

Komutların yazılış biçimlerini açıklarken bazı tanımlama harfleri kullanılır.

f: file register

d: destination (hedef)

d=0 → W kaydedici

d=1 → file register

d parametresinin kullanıldığı yerde 0 veya 1 yazmak hatırlatıcı olmayabilir. Bunu dikkate alarak editör ve yorumlayıcılar (hazırladığımız editör ve derleyicide de bu izin vardır) 0 yerine w, 1 yerine f yazmaya izin verir.

k: sabit veya adres etiketi

b: kullanıldığı komuta göre bit tanımlayıcı yada binary sayıları belirleyen harf (örneğin b"00001111" gibi)

d: desimal sayıları belirleyen harf (örneğin d"16" gibi)

h: hex sayıları belirleyen harf (örneğin h"EF" gibi)

Byte-Yönlendirmeli Komutlar: Komut f,d

File Kaydedici (f): Hexadesimal adres veya file kaydedici adı

Hedef (d): Komutun çalışmasından sonra sonucun nereye yazılacağını belirler.

d=0 → W kaydedicisi

d=1 → f ile belirtilen kaydedici

Bit-Yönlendirmeli Komutlar: Komut f,b

File Kaydedici (f): Hexadesimal adres veya file kaydedici adı

Bit tanımlayıcı (b): 0-7 arasında onaltılık sayı veya EQU ile tanımlı etiket

Sabit İşleyen Komutlar: Komut k

(k): Sabit

Örnekler: H"EF" / D"16" / B"00001111"

Kontrol Komutları: Komut k

(k): Adres etiketi

### 3.6.1. Adresleme yöntemleri

PIC16F84 doğrudan ve dolaylı olmak üzere iki farklı adresleme moduna sahiptir. Veriye daha hızlı erişmek açısından genellikle doğrudan adresleme yöntemi tercih edilir.

Doğrudan Adresleme: Veri hafıza alanı içerisinde herhangi bir dolaylı adresleme kaydedicisi kullanmadan gerçekleşen adresleme türüdür. Bu tip adresleme modunda komutların karşısında bulunan kaydedicilere doğrudan veri yüklemesi yapılır.

MOVLW H"EF"; Doğrudan sabit EFh bilgisi W kaydedicisine yüklenir.

MOVWF TRISB; W kaydedicisinin içeriği doğrudan RAM hafızada 86h adresinde bulunan TRISB kaydedicisine yüklenir.

BCF TMR0,2; Doğrudan RAM hafızada 01h adresinde bulunan TMR0'ın 2. biti sıfırlanır.

Dolaylı Adresleme: Dolaylı adreslemede, doğrudan adreslemede olduğu gibi hafıza alanına bir komut satırında ulaşamaz. Bunun için ek kaydedicilere ihtiyaç vardır. Dolaylı adreslemede kullanılan bu kaydediciler INDF ve FSR kaydedicileridir.

INDF kullanılarak yapılan adreslemede FSR kaydedicisi dosya gösterici olarak görev yapar.

MOVLW h"25"; 25h bilgisi W kaydedicisine yüklenir.

MOVWF FSR; W kaydedicisinin içeriği FSR kaydedicisine yüklenir. Bu işlem gerçekleşir gerçekleşmez aynı zamanda FSR'ye atanan 25h bilgisi RAM hafızadaki 25h adresini temsil eder ve bu adresteki bilgiyi INDF kaydedicisine yükler.

MOVF INDF,0; INDF kaydedicisinin içeriği W kaydedicisine yüklenir. Böylelikle dolaylı olarak 25h adresindeki bilgi W kaydedicisine yüklenmiş olur.

PIC16F84 mikrodenetleyicisinin tüm komutlarının işkodları, kaç saykılada işlendikleri, etkiledikleri bayraklar, kullanımları ve örnekler Ek-C'de verilmiştir.

## **BÖLÜM 4. SİMÜLASYON**

Statik resim, canlandırma ve dijital videolar öğrenilecek kavram ve süreçlerin bir kopyasını bilgisayar ekranına aktarırken, diğer bir görsel gösterim biçimi olan simülasyonlar öğrenilecek konuya ait dünyanın bir modelini öğrenciye sunar. Öğrenci bu modele ait değişkenleri farklı değerler vererek çalıştırır ve sonuçlarını inceler. Simülasyon modelleri konuya göre bir denklemler sistemi, bir yöntemler seti veya bir neden sonuç seti olabilir. Bilgisayar simülasyonlarını canlandırmalardan ayıran özellik işte bu noktada kendini göstermektedir: Etkileşim [24].

Öğrenme ortamında karmaşık konu örüntülerinin basite indirgenerek çalışılmasına olanak veren simülasyonlar zaman tasarrufu sağladıkları gibi, kaynaklarda ekonomiklik de sağlar [24].

Bu çalışmada 8085 mikroişlemcisi ve PIC16F84 mikrodenetleyicisi için gerçekleştirilen simülasyonları geliştirmede yaygın kullanıma sahip Macromedia Flash programı kullanılmıştır.

### **4.1. Sanal Laboratuvarlar**

Fen bilimleri içeriğinin (sayısal konuların) genelde soyut yapı taşları içermesi, bu alanda yaparak, yaşayarak, etkinliklerle dolu bir öğretimi zorunlu hale getirmektedir. Bu bağlamda laboratuvar, öğrencilerin deneyim kazanacağı eğitimin önemli bir bileşenidir. Bununla birlikte, malzeme eksikliği ve laboratuvar yetersizliği gibi nedenlerle sınırlı tutulan öğrenci çalışma saatleri, çoğu zaman deneylerin kalabalık gruplar halinde yada gösteri deneyi formatında gerçekleştirilebilmesini mümkün kılmaktadır. Bu durum, bilginin bireysel deneyim ve gözlemle oluşturulabileceğini savunan laboratuvar yönteminin temel felsefesine aykırı düşmektedir. Geleneksel

yöntemlerin bu tür kısıtlamaları göz önüne alındığında uygun alternatiflerin aranma zorunluluğu ortaya çıkmakta ve bilgisayar temelli sanal laboratuvarlar, geleneksel laboratuvarlara bir destekçi olarak büyük bir potansiyel kazanmaktadır [25].

Sanal laboratuvar terimi; birincisi laboratuvar yapı ve işleyişinin bilgisayar simülasyonunun gerçekleştirilmesi, diğeri de laboratuvar donanımına uzaktan erişim olmak üzere iki anlamda kullanılmaktadır. Bu kapsamda sanal laboratuvarlar iki sınıfa ayrılabilir:

Laboratuvar yapı ve işleyişinin simülasyonu: Laboratuvarların yapı ve işleyişinin uygun programlar yoluyla bilgisayar ortamında yapay olarak oluşturulmasına dayanan simülasyon uygulamalarıdır.

Laboratuvar donanımına uzaktan erişim: Uzaktan erişimli (Internet destekli) laboratuvarlar, farklı coğrafi mekanlardaki kullanıcıların uzaktan laboratuvar donanımına erişimi ve komut gönderip geri bildirim bilgisi ve ortam görüntüsü almasının sağlandığı uygulamalardır [26].

Bu tez çalışmasında kullanılan sanal laboratuvar birinci tip yapıdadır. Bu tez çalışmasında ortaya çıkan simülatör yazılımında öğrenciler çalışmalarını gerçek bir fiziki mekan bulunmadan ister kendi bilgisayarlarında, isterlerse de ağ veya internet ortamındaki başka bir bilgisayar üzerinden yapabilmektedirler.

Geleneksel laboratuvarlar ve sanal laboratuvarların karşılaştırılması: Pratik eğitimle basit G/Ç işlemlerinin programlanmasının öğrenilmesi çok önemlidir. Her ne kadar simülatörde G/Ç elemanlarının gerçek fonksiyonlarının tam olarak taklit edilmesi zor olsa da, pahalı gerçek cihazların üzerinde denemeler yapmak yerine, simülatör türü sanal programlar üzerinde uygulamalar geliştirmek daha uygundur. Simülatörler üzerinde denemesi yapılan programlar başarılı olduktan sonra gerçek donanımlara uygulanabilir. Yanlış uygulamalardan dolayı oluşabilecek hatalar, simülatörler üzerinde kolayca giderilebilmektedir [14].

Tipik bir geleneksel laboratuvar; cihaz veya gereçler üzerinde doğrudan çalışan



öğrenciler ile laboratuvar araçlarını ve prosedürlerini içerir. Laboratuvarlarda çalışmalar, genellikle eğiticiler, laboratuvar teknisyenleri ve/veya öğretmenlerin kontrolü altında yapılır. Geleneksel laboratuvarlarda birçok farklı problem ortaya çıkar [6]. Bunlar:

- Maliyet: Cihazlar, depolama ve bakımın maliyeti yüksektir
- Kullanım Sınırlaması: Öğrencilerin genellikle sadece laboratuvar saatleri sırasında olmak üzere laboratuvarlara erişimi sınırlandırılmıştır.
- Yetersiz Eğitim: Bazı durumlarda laboratuvarların durumlarından dolayı, hatalı sonuçlar ortaya çıkmaktadır. Ölçülen veriyi el ile işlemek ve aynı ölçümleri sıkıcı prosedürler ile tekrarlamak önemli bir zaman almaktadır. Sonuçları tartışabilmek için deneylerin sonucunda az bir zaman dilimi kalmaktadır.
- Güvenlik: Öğrencilerin deney sırasında yaygın bir biçimde deney sırasında tehlike yaratabilecek cihaz ve araçlarla çalışması durumu ortaya çıkabilmektedir [6].

Benzetimli laboratuvarlarda bilgisayarlar, geleneksel laboratuvarlarla tamamen yer değiştirerek laboratuvar cihazlarının çalışmasını modeller ve simüle eder. Modelleme ve simülasyonlar geleneksel laboratuvar sistemlerinin eşdeğer matematiksel modelleridir. Tipik olarak dördüncü nesil diller kullanılarak daha kompleks simülasyonlar yazılabilirken, Delphi veya Java gibi üçüncü nesil diller kullanılarak daha basit simülasyonlar yazılabilir. İyi tasarlanmış sanal laboratuvarlar ile elde edilen avantajlar aşağıdaki gibi sıralanabilir [27] [28]:

- Maliyet Verimliliği: Sanal laboratuvarlar laboratuvar gereçlerinin ve kullanılan sarf malzemelerin bir kısmı veya tamamının yerini alabilir. Bundan dolayı cihaz ve sarf malzemelerin satın alma maliyetleri, bakım ve depolama maliyetlerini oldukça düşürülebilir.
- Kullanılabilirlik: Bilgisayarlar ve bilgisayar ağları, herhangi bir zamanda ve herhangi bir yerde laboratuvar hazırlamak için yardımcıdır. Bu sayede istenilen yer ve zamanda kısıtlama olmaksızın kullanıma hazır hale getirilebilir.

- Aktif (etkin) Öğrenme: Çoğu prosedürler için (ölçme ve formatlama gibi), öğrenciler üzerine yükümlülük getiren işleri azaltmak bilgisayarlar kullanılabilir. Böylece zaman tasarrufu sağlanır.
- Güvenlik: Bilgisayarlar potansiyel tehlike durumlarının engellenmesini sağlayarak, öğrencileri oluşacak tehlikelerden korur. Öğrencilerin laboratuvar cihazı ile doğrudan etkileşimi azaltılabilir veya yok edilebilir.
- İdari Faydalar: Sanal laboratuvarlar not, kayıt tutma ve geri besleme gibi çevrimiçi değerlendirmelerde öğreticilere yardımcı olabilir.

Bir kısım teknolojiler gibi sanal laboratuvarlarda simülasyonların gerçekçi olmayan doğallığı, öğrenci kontrolünün eksikliği, vb. dezavantajlara sahiptir [6].

#### 4.2. Macromedia Flash

Macromedia firması 1997 yılında 'Future Splash Animator' adlı programı satın alarak grafikler oluşturmak, bu grafikleri hareketlendirmek, grafikleri tarayıcıda ve web sayfası olarak gösterebilmek için gereken araçları bir arada bulunduracak şekilde geliştirdi ve 'World Wide Web' için grafiksel içerik oluşturan bir araç olarak 'Flash' programı ismi ile piyasaya sunuldu. 'Flash' programı 1997 yılından günümüze kadar gelişim gösterdi ve farklı isimler ile farklı sürümler şeklinde (Flash 2.0, Flash 3.0, Flash 4.0, Flash 5.0, Flash MX vb.) piyasaya sunuldu. 2006 yılında video ve resim işleme özellikleri güçlendirilerek Flash 8.0 sürümü piyasaya sunuldu.

Flash'ın en önemli özelliklerinden birisi, görüntü teknolojisinin temelini oluşturan vektörel grafiklerdir. Vektörel grafiklerde, grafik matematiksel ifadeler halinde saklanır. Vektör komutları, grafiği bir dizi çizgi ve yay olarak tanımlar. Flash programı ile hazırlanan animasyonlarda çizilen nesnenin hareketi de matematiksel ifade olarak saklanacaktır. Bu durumda çizilecek bir şeklin kenar uzunluğu, hareketin nerde başlayıp nerde biteceği ve hareketin ne kadar zamanda tamamlanacağı animasyon boyutunu etkilemeyecektir. Herhangi bir büyüklükteki kare şeklinde çizimin bir konumdan başka bir konuma hareket ettirildiği animasyonun dosya boyutu 1KB'ı geçmeyecektir. Çözünürlüğün artması yada grafiğin büyütülmesi durumunda çizim program tarafından otomatik boyutlandırılacaktır ve görüntü

kalitesi bozulmayacaktır.

Bitmap grafikler ise her nokta konum ve renk bilgisi olarak nokta-nokta saklanır. Bu yüzden dosya boyutu büyüktür. Örneğin 2.5cm'lik kare şeklindeki bir çizim 72 dpi çözünürlükte yaklaşık 25KB dosya boyutuna sahip olacaktır. Resmin boyutu cm olarak artırılırsa ya da resim yüksek çözünürlükte kaydedilirse dosya boyutu katlayarak artacaktır.

Bitmap grafiklerde görüntüleme amaçlı kullanılan herhangi bir programda grafik yakınlaştırıldığında görüntüleme programı tolare etmeye çalışsa bile görüntü kalitesi bozulmaktadır.

Flash'ın eş zamanlı yükleme ve oynatma özelliği sayesinde, vektörel bilgiler aynı anda hem yüklenip hem de yüklenen kısımlar gösterilebilir. Flash ürünü içine gömülen ses ve video dosyaları olduğunda bu özellik gerçekten faydalı olacaktır. Aksi durumda ses ya da video dosyasının oynatılmak üzere belleğe tamamen indirilmesi beklenecekti ki dosya büyüklüğüne bağlı olarak bu dakikalar hatta saatler alabilecek bir süreçtir.

Flash programının Actionscript denilen kendine özgü bir script dili vardır. Bu script dili kullanılarak hazırlanan animasyonlar istediğimiz sırada ve düzende oynatılarak etkileşimli hale getirilebilmektedir.

Flash ürünü kullanabilmek (izlemek) için Flash oynatıcı olması gereklidir. İşletim sisteminden bağımsız olarak tüm internet tarayıcıları Flash ürünleri izletecek eklentilere sahiptir. Flash ürün dosyası swf dışında exe, mov, avi, gif, jpeg gibi birçok yaygın dosya formatında kaydedilebilmektedir. Exe dosya formatında kayıt edilmiş bir Flash ürün dosyasını çalıştırmak için Flash oynatıcı gerekmemektedir.

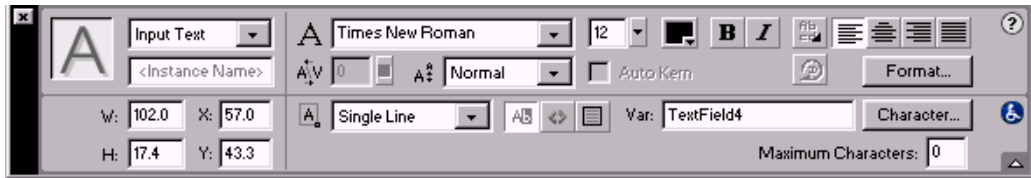
Flash programının aşağıda sıralanan eksiklikleri bulunmaktadır:

- Veri tabanlarına erişememekte, kayıt, düzenleme ve silme işlemleri yapamamaktadır.

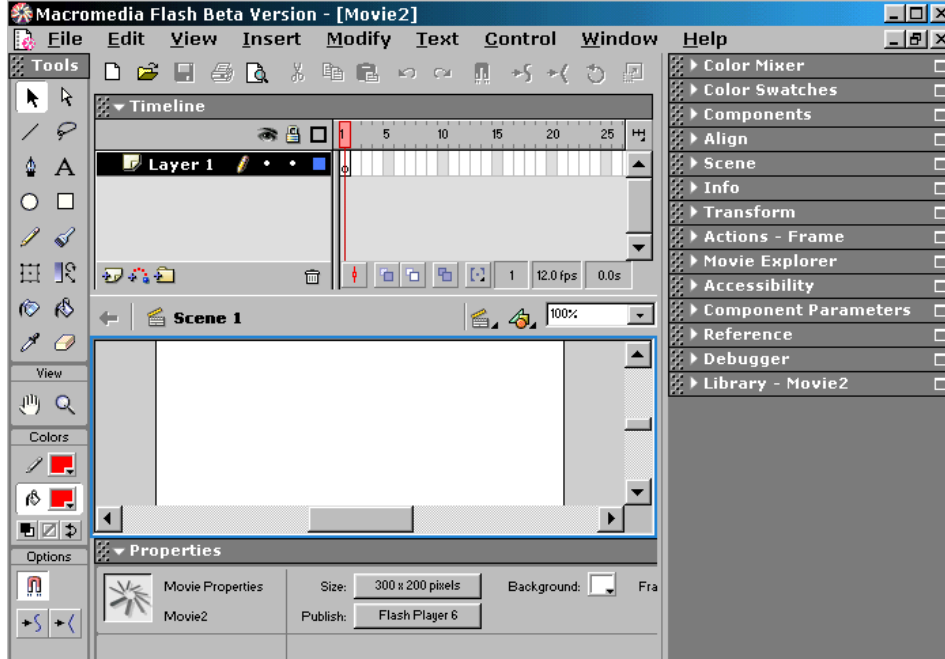
- Adı ve adresi verilen dosyaya ulaşabilmekte ancak mevcut dosya sistemindeki klasörler ve içlerindeki dosyaların listesine ulaşamamaktadır.
- Harddisk üzerindeki metin tabanlı dosyaları okuyabilmesine rağmen metin tabanlı dosya oluşturamamaktadır.

Flash MX ile sunulan yeni özellikler aşağıdaki şekilde özetlenebilir:

1. Çok fazla katmanlı çalışmalarda istenilen katmanları tek bir dosya katmanı şeklinde toplanabilmekte ve düzenlenebilmektedir.
2. Tüm sembollerin özellikleri, metin ve metin kutuları, tween işlemleri properties penceresinde düzenlenebilmektedir (Şekil 4.1).
3. Paylaşılmış kütüphanelerde daha kolay güncelleme ve sembol değişimi yapılabilmektedir.
4. Semboller ve metinlerin break apart yöntemi ile sembol özellikleri kırıldıktan sonra her birim bu komutla ayrı katmanlara dağıtılabilmektedir.
5. Macintosh OS X, Windows XP, Mac OS 9.1 ve üstü, Windows 95, 98, 2000, ME ve NT desteği vermektedir.
6. Araç kutusunda skew ve distort eklentileri bulunmaktadır.
7. Envelope aracı eklentisi ile grafiklerin şekillerinde değişiklik yapılabilmektedir.
8. Actionscript ile maskeler yapılabilmekte ve kontrol edilebilmektedir.
9. Yakınlaştırma işlemlerinde sahnede pikselleri gözlememiz mümkün olmaktadır.
10. Video Desteği: AVI, MOV, DV, MPEG formatında video çalışmaları swf dosyasına gömülerek kullanılabilir.
11. Harici MP3 ve JPG dosyalarının flash içine yüklenilebilmektedir.
12. Dikey düzlemde metin yazılabilmektedir.
13. Hazır şablonlar ve componentler çalışmaya eklenebilmektedir [29].



Şekil 4.1 FlashMX 2004 Text kutusu için properties paneli



Şekil 4.2 FlashMX 2004 genel pencereler görünümü

### 4.3. Mikroişlemci – Mikrodenetleyici Simulasyonları

Simülatörler genel olarak gerçek cihazlar kullanmadan sanal bir ortamda gerçek olayın taklidini yaparlar. Simülasyonu yapılacak olan işlemci farklı bir işlemci belleğinde canlandırılıp zamandan ve maliyetten tasarruf edilir [30].

Assembly dilinde yazılan bir programın makine kodlarına çevrildikten sonra, test edilmesi için çeşitli yöntemler vardır. Bu yöntemlerden ilki, programın işlemciye uygun bir simülatöre girdi olarak verilmesidir. Diğer, programın sahip olduğu işlemciyi kullanan bir sistemde çalıştırılmasıdır. Son olarak bir mikroişlemci geliştirme setinin parçalarından biri sayılan bir emülatörde programın test edilmesidir. Her geliştirme aracı kendine has amaçlarla kullanılmasına rağmen, ortak olarak kullanıcının programı üzerinde düzeltmeler yapmasını kolaylaştıracak yeteneklere sahiptir. Bu yetenekler genellikle, programın istenilen bir bölümünün çalıştırılması, komutların çalışmasının izlenmesi, bellek üzerinde bir bölgenin görüntülenmesi ve işlemcinin sahip olduğu kaydedicilerin içeriklerinin izlenmesi gibi fonksiyonlardır [31].

Mikroişlemci simülasyonu, belirlenen mikroişlemci için yazılan programın farklı mikroişlemciye sahip bilgisayar üzerinde çalıştırılması işlemidir. Simülatörü gerçekleştiren mikroişlemci mimarisi, farklı sistemin belleğinde tanımlanmış bir bellek bloğunda tekrar oluşturulur. Mikroişlemcinin sahip olduğu bellek ve kaydediciler farklı işlemcinin belleğinde temsil edilirler. İşlemcinin komut setindeki her komut diğer makinede birkaç komutla gerçekleştirilir. Simülasyon sırasında, farklı işlemcideki bellek bloğunda adeta programın yazıldığı işlemcide çalışıyormuş gibi değişiklikler yapılır. Bu şekilde kaydedicilerde programa uygun olarak gerekli değişiklikler yapılmış olur [31].

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 geliştirilmesinde de kullanılmasını gerekli kılmaktadır. Simülatörün bir yazılım geliştirme aracı olarak kullanılması amaçlandığında aşağıda sıralanan bazı kısıtlamalar ortaya çıkmaktadır [14]. Bunlar;

- Simülatör, mikroişlemcinin 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 mikroişlemci mimarisini ve assembly dilini öğretmek için geliştirilen 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 tesbit edilemezler.

Tüm mikroişlemci ve denetleyiciler için yazılan simülatörlerde standart olarak editör assembler ve hata ayıklayıcılar bulunur.

#### **4.3.1. Assembler tasarımı**

Assembler, bir metin editöründe assembly dili kurallarına göre yazılmış olan

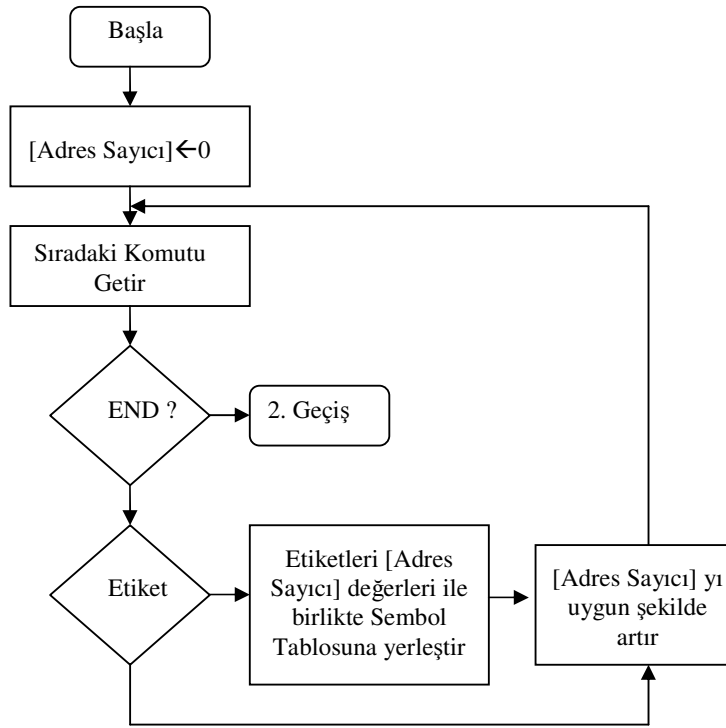
komutları mikroişlemci/mikrodenetleyicinin anlayabileceği (makine diline yakın bir formata) hexadesimal kodlara çeviren (derleyen) bir programdır [19]. Makine kodu üretmek için assembler, ADC, LDA ve JMP gibi assembly komutlarını alır (kaynak program) ve bunları bir dizi ikilik kodlara (amaç program) çevirir. Ayrıca programcı tarafından tasarlanan etiketlenmiş makine kodu adreslerini gerçek makine bellek alanlarına dönüştürür [32].

Assembler, komut ve işlenecek kısımları komutun yapısına göre hex kod düzeninde işkodu (opcode)'na çevirir. Assembler, programın işlemci tarafından doğrudan çalıştırabilmesi için kaynak programdaki sembolik adreslerin yerine gerçek adresleri koyar.

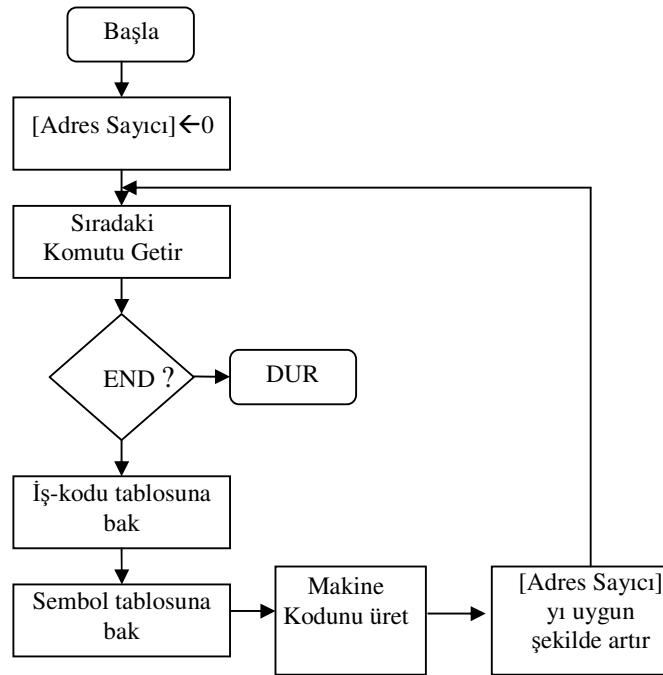
Assembler programları geliştirilene kadar assembler işlemlerinde, programcı hedef mikroişlemci / mikrodenetleyicinin komut kümesindeki kodlarına bakarak doğrudan işkodlarını üretmekteydi ki bu çok zaman almaktaydı. Assembler programları ile birlikte program geliştirme işlemleri hem kolaylaşmış hem hızlanmış.

Assembly dili programcının hex adresler yerine sembolik adresler kullanabilmesine imkan tanır. Derleme esnasında çağırma komutlarının (CALL vb.) gösterdiği sembolik adres hex adrese çevrilirse işkodu üretilebilir. Bu nedenle derleme işlemi iki geçişte yapılır. Birinci geçişte doğrulanan her satır için hex adres hesaplanır ve etiketler bir dizide biriktirilir (Şekil 4.3). İkinci geçişte sembolik adresler yerine hex adresler yerleştirilerek tüm komut satırları onaltılık düzende işkoduna çevrilir (Şekil 4.4) .

Assembly dilinde yazılan programların makine diline çevrilmesi sırasında, programcı tarafından gerekli kurallara uyulmadığı takdirde ortaya çeşitli hatalar çıkmaktadır. Bu hatalar, komut kümesinde bulunmayan ve başka işlemcilere ait komutların veya anlamsız mnemonic dizisi girilmesi halinde ortaya çıkabileceği gibi, yine assembler tarafından tanımlanmamış bazı talimatların girilmesi durumunda da ortaya çıkacaktır [14].



Şekil 4.3 Assemblerin birinci geçiş algoritması



Şekil 4.4 Assemblerin ikinci geçiş algoritması



### 4.3.2. Hata ayıklama ve test etme

Assembly dilinde program yazmanın zorlukları, diğer yüksek düzeyli dillere nazaran daha fazladır ve hata yapma ihtimali oldukça yüksektir [14].

Derleme adımı ilk ve kolay aşamadır. Programcı derleme aşamasından geçen programının hedefine uygun olarak işleyip işlemediğini kontrol etmek ister. Bir mikroişlemci sistemin hata ayıklama ve test işlemi istem tasarımının en önemli kısmıdır. Hata ayıklama, programın yapısındaki sorunların yok edilmesi veya düzeltilmesi işlemidir. Çünkü program içerisinde ortaya çıkabilecek sorun oldukça karışıktır ve tahmin edilemez. Bu nedenle programı oluşabilecek hatalardan arındırmak için debugger (hata ayıklayıcı) denilen program araçları geliştirilmiştir.

Mikroişlemci / mikrodenetleyici hata ayıklama araçlarının yapabilecekleri işlemler:

- Programda kritik satırlarda durma noktası koyabilme,
- Seri çalışma dışında adımlayarak çalışma,
- Çalışmayı durdurma,
- Kaydedici, bellek ve PIA port içeriklerini sıfırlayarak programı yeniden başlatma,
- İstenilen noktada kaydediciler ve bellek içeriklerinin dökümünü görebilme,
- Hata sezmesi durumunda kullanıcıya programa kalındığı yerden devam edebilme,
- Kaydedicilerin içeriklerini inceleme ve değiştirme,
- Bellek içeriklerini inceleme ve değiştirme,
- PIA portlarının içeriklerini inceleme imkanı vermesi şeklinde sıralanabilir.

Mikroişlemcili sistemlerde hata ayıklama işlemi, doğrudan kaydedici içeriklerini incelemek ve donanım ile yazılım arasındaki hareketlere daha yakın olmak ve gerçek zamanlı uygulamalardaki hareketleri anında ayrıntılı olarak görebilmek için yavaş (adım adım) çalışma yeteneğine sahip olmalıdır [14].

## BÖLÜM 5. 8085 SİMÜLATÖRÜ TASARIMI ve UYGULAMASI

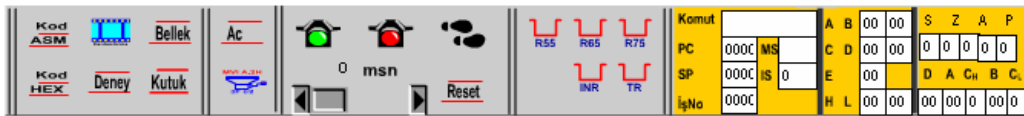
### 5.1. Genel Tasarım

8085 Simülatörü, assembly dilinde program yazılabilecek bir metin editörü (düzenleyici), yazılan bu programı derleyebilecek bir assembler, derlenen programın hatalarını bulacak ve ayıklayabilecek bir debugger, işlemci içerisindeki olayların gözlemlendiği simülatör ve yazılan programlara uygun olarak çalışan uygulama bölümlerinden oluşmaktadır. Tüm bu bileşenler, internet tarayıcı yada Flash player bulunan tüm işletim sistemlerinde çalışmaktadır.

Simülatör üzerinde gerçekleştirilen tüm işlemler hazır komutlar yerine ilgili butonlara fare ile tıklamak suretiyle yapılmaktadır.

Araç çubuğu penceresinin sol kısmında komut editörü, bellek, animasyon, deneyler, işlem kütüğü ve programın derlenmiş penceresini açma/kapama butonları bulunmaktadır (Şekil 5.1). Bu butonlara bir kere basıldığında pencere açılmakta ikinci basıldığında pencere kapatılmaktadır.

Araç çubuğunda pencere açma/kapama butonlarından sonra sırayla yazılan programı derleme ve çalıştırma ile ilgili butonlar, kesme butonları ve son işlenen komut satırının kaydediciler üzerindeki etkisinin görülebileceği bir ekran bulunmaktadır. Araç çubuğunun sağında yer alan kaydedici içeriklerini gösteren kısımdan aynı zamanda kaydedici değerleri de kullanıcı tarafından değiştirilebilmektedir.



Şekil 5.1 8085 simülatörü araç çubuğu

## 5.2. Editör Tasarımı

Simülatörde kullanılan metin düzenleyicisi (editör), 8085 mikroişlemcisi için uygun komutlar kullanılarak assembly dilinde program yazılmasını sağlayan simülatörde yerleşik bir araçtır. Yazılan program araç çubuğu üzerinde bulunan assembler butonuna basılarak derlenebilir. Editör penceresi üzerinde işletim sisteminin desteklediği kes, kopyala ve yapıştır gibi düzenleme işlemleri klavye kısayolları yada farenin sağ tuş menüsü seçenekleriyle yapılabilmektedir. Programın sabit diske dosya olarak kaydedilmesi editörün yazıldığı Flash programının yetenekleri dahilinde olmadığından program simülatörden başka bir metin editörüne kopyalanarak kaydedilebilir.

Editörde program yazarken, programın derlenmesi aşamasında istenmeyen hatalar ortaya çıkmaması için assembly dilinin yazım kurallarına uyulması zorunludur. Editör penceresinde assembly dilinde program yazarken dikkat edilecek kurallar aşağıdaki şekilde özetlenebilir:

- Assembly dilinde yazılan her komut satırı sırayla etiket, iş-kodu (operatör-komut), işlenen (operand) ve açıklama işlem alanlarından oluşmaktadır.
- İşlem alanları birbirinden boşluk veya “,” karakteriyle ayrılmalıdır.
- Gereğinden fazla boşluk konulmamalıdır. Aksi takdirde programın okunabilirliğini azaltmaktadır.
- Komuta ait işlenenler birbirlerinden “,” karakteriyle ayrılmalıdır.
- İşlenen alanında komuta ait parametre değer ifade edecekse ilk karakteri sayı sistemini belirten H (hex-onaltılı) , B (binary-ikili) veya D (decimal-onluk) harflerinden biri olmalıdır. Değer çift tırnak arasında yazılmalıdır. Örneğin H"EF".
- Etiket olarak seçilen isimlerin ilk karakteri hiçbir zaman sayısal bir değer olamaz. İlk karakter harf seçildikten sonra diğerleri sayısal bir karakter olabilir.
- Etiket isimlerinde noktalama işaretleri gibi işaretler olamaz. Etiket ismi harf ve sayılardan oluşur.
- Aynı etiket iki kere kullanılamaz (derleyici ilk etiketi geçerli kabul edecektir) ancak birden fazla işlenen alanında kullanılabilir.

- Programın okunabilirliğini arttırmak için gerekli açıklamalar, komut satırının bitiminden başlar. Dikkat edilmesi gereken nokta, assemblerin bu açıklamaları göz önüne almamasını sağlayacak noktalı virgül (;) işaretinin hemen açıklamaların başına konulması gerekliliğidir.
- Kullanıcı atama deyimi EQU ile programda kullanmak üzere kendi sabitlerini tanımlayacaksa bunu program bloğundan önce yapmalıdır. Aksi durumda kullanıcının tanımladığı sabitler program bloğunda dikkate alınmayacak ve assembler işleminde o program bloğundan sonra tanımlanmış bu sabitleri içeren komut satırları derlenemeyecektir.
- Komut kodlarının belleğe yerleşeceği adres, komutlardan önce kullanılacak ORG komutu ile belirlenir. Örnek: ORG H"0200" satırından sonra gelecek komutlar belleğe H"0200" adresinden yerleşmeye başlayacaktır.

Metin editöründe komut satırı tamamen büyük harfe çevrilerek derlendiğinden etiket, komut yada işlenenlerde büyük yada küçük harf kullanılması fark etmemektedir.

Assembly dilinde satırda sadece etiket varsa tek etiket olan satırdan sonra gelen satır komut ile başlıyorsa etiket o satırı göstermektedir. Yazılan editörde satırdaki tek kelime etiket kabul edilmesine rağmen sonraki satıra ait olarak kabul edilmemektedir.

Yazılan editörde Assembly dilinde komutları yazılırken çağırma komutları ile birlikte etiket kullanılmaktadır. Programlama dillerinde etiketin kullanılma sebepleri şunlardır;

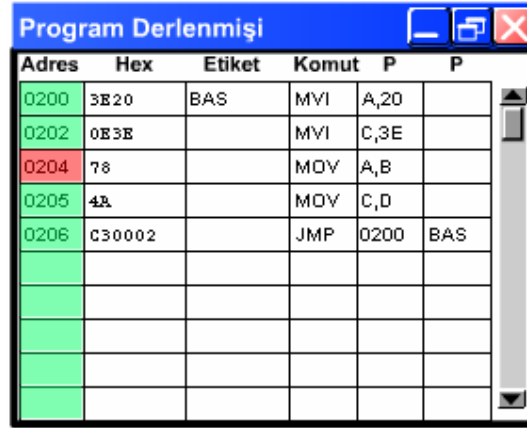
- Program alanının kolaylıklar bulunmasının ve hatırlanmasını sağlar,
- Programda bir değişiklik veya yeniden düzenleme yapıldığında, belirlenen adresler kayacaktır ancak etiketlenmiş blok önceki belirlenen aynı adresi korumuş olacaktır.

### 5.2.1. Derleyici tasarımı

Bu bölümde, simülatör içerisinde yer alan 8085 assemblerinin ayrıntılı tasarım ve

yürütülme aşamaları ele alınmaktadır.

Hex kodlar haline getirilen program, gerçek bir mikroişlemcide veya bir simülatörde çalıştırılabilir. Assembler işlemi sonunda elde edilen assembly kodlarla birlikte onaltılık düzende makine kodları ve bu kodlara karşılık gelen adreslerin yer aldığı Programın Derlenmiş penceresi Şekil 5.2’de sunulmaktadır:



Adres	Hex	Etiket	Komut	P	P
0200	3E20	BAS	MVI	A,20	
0202	0E3E		MVI	C,3E	
0204	78		MOV	A,B	
0205	4A		MOV	C,D	
0206	C30002		JMP	0200	BAS

Şekil 5.2 Derleme işlemi sonuç listesi ve durma noktaları seçimi

Programın hex kod düzeninde işkodu haline getirilmesi için hedef mikroişlemcinin kullanılmasına gerek yoktur.

Derleme işleminde yapılanlar:

- Program chr(13)-Enter karakterine göre satırlara ayrıştırılmaktadır.
- Satırlar sırayla (“;” karakteri öncesi yani açıklamaya kadar olan kısımları) satır kontrol işlemine tabi tutulmaktadır.
- Satır kontrol işlemlerinde program bloğundan önce atama bloğundaki “EQU” etiketli satırlar kullanıcı sabiti olarak ayrı bir diziyeye kaydedilmektedir.
- Satır kontrol işleminde değerlendirilen satırlardan hatasız olanlar çalıştırılacaklar dizisinde, dallanma ve atlama komutlarının çağırdıkları etiketler ve buldukları satırın numarası çağıran dizisinde, satırların etiketleri ise etiket dizisinde tutulmaktadır.

- Çağırın dizisinde tutulan etiketler sırayla etiket dizisiyle karşılaştırılmakta ve karşılığı olmayan etikete sahip satırlar çalıştırılacaklar dizisinden silinmektedir.
- Çalıştırılacaklar dizisinden silinmiş satırlar olacağı için etiketlerdeki satır numaraları tekrardan düzenlenmektedir.
- Çağırınlar için son kez etiketlere karşılık gelen hex kodları düzenlenmektedir.

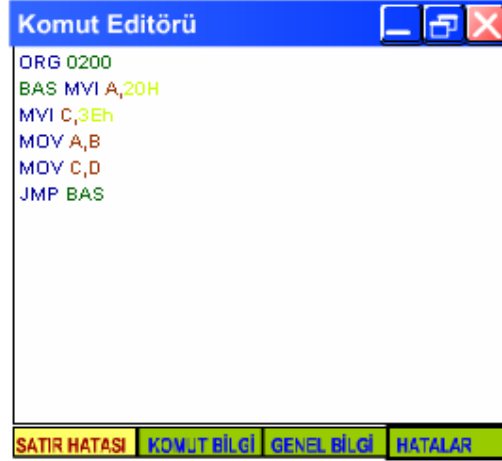
Satır kontrolü işleminde yapılanlar:

- Yazılan satır boşluk, “,” ve “;” karakterlerine göre kelime ve gruplara ayrıştırılır. Kelime ayrıştırma işleminde beş farklı alan kullanır: komut, kaydedici, etiket, değer ve açıklama.
- Ayrıştırılan kelime, kelime alanının çeşidine göre renklendirilmektedir. Bu işlem kodları okumayı ve hataları görebilmeyi kolaylaştırmaktadır.
- Ayrıştırılan kelimeler kelime, tür ve değer olarak bir dizide biriktirilmektedir. Bu dizideki kelimeler genel satır değerlendirmesinden sonra hatasız ise parametre değerlendirmesinden geçirilmektedir. Hatalı durumlar kullanıcıya bildirilmekte ve satır derlemeye dahil edilmemektedir.
- Genel satır değerlendirmesinde assembly dili yazım kurallarına uygunluk kontrol edilmektedir.
- Parametre kontrolünde satırda bulunan komuta uygun sayıda ve nitelikte işlenen kullanılıp kullanılmadığı kontrol edilmektedir.
- Derleme yapılıyorsa hatalı satırlar hata numaralarıyla birlikte bir diziye kaydedilmektedir.
- Seçime göre metin kutusunda imlecin üzerinde bulunduğu satıra ait hata ya da satırda kullanılan komutun kullanımı ile ilgili bilgi, araç çubuğunun sağ tarafında yer alan bilgi kutusunda görüntülenmektedir.

### 5.2.2. Derleme hata mesajları

Editör penceresinde yazılan programın makine diline çevrilmesi için Assembler butonuna basıldığında, seçime göre programdaki hatalı tüm satırlar ile ilgili araç çubuğundaki bilgi kutusunda hata mesajları verilmektedir.

Editör penceresinde imlecin üzerine hareket ettiği satır hata kontrolünden geçirilerek hata olması durumunda kullanıcıya bildirilmektedir (Şekil 5.3).



Şekil 5.3 Komut editörü

Programcının yazdığı kodların makine diline çevrilmesi bir dizi kurallar çerçevesinde gerçekleşir. Bu kurallar çeviriciyi (assembleri) yazan kişilerce belirlenir. 8085 mikroişlemci simülatörüne has assemblerde oluşabilecek hatalar satır ve parametre hataları olmak üzere ikiye ayrılmaktadır. Bu başlıklar altında kullanıcıya verilen hata mesajları aşağıda verilmektedir.

Satır hatası mesajları:

1. Bir komut satırı en fazla 4 kelime olmalıydı
2. İlk kelime komut yada etiket olmalıydı
3. Bir komut satırında yalnızca bir tane komut olur!
4. CALL-JMP ve türevleri dışında komuttan sonra etiket olamaz!
5. İlk iki kelimedenden biri mutlaka komut olmalıydı.
6. İlk kelime komut olduğundan satır 3 kelimedenden fazla olamaz!

Parametre hatası mesajları: Komutlar kullandıkları parametre yapısına göre parametresiz, bir parametrelili ve iki parametrelili olarak üçe ayrılmaktadır. Kontrol edilen satırda parametre hatası araç çubuğundaki bilgi kutusunda programcının doğru kullanımını sağlayacak bilgilendirme olarak verilmektedir.

Parametre hatası mesajları:

1. Her iki parametreside yok ancak olmalıydı
2. Veri bilgisi hex sayı olarak iki basamaktan fazla olmamalıydı
3. Adres bilgisi hex sayı olarak dört basamaktan fazla olmamalıydı
4. Parametre 0-7 arası bir değer olmalı
5. Birinci parametre yok
6. İkinci parametre olmamalıydı
7. Bu komut için parametre kullanılmaz
8. Bu komut için ilk parametre A-B-C-D-E-H-L-M kaydedicilerinden biri olmalı
9. Bu komut için ilk parametre B-D kaydedicilerinden biri olmalı
10. Bu komut için ilk parametre B-D-H-SP-M kaydedicilerinden biri olmalı
11. Bu komut için ilk parametre B-D-H-PSW kaydedicilerinden biri olmalı
12. Bu komut için ilk parametre B-D-H-SP kaydedicilerinden biri olmalı
13. Bu komut için ikinci parametre A-B-C-D-E-H-L-M kaydedicilerinden biri olmalı

### 5.3. Hata Ayıklama ve Test Etme

Bu noktaya kadar yapılan işlem, editör kullanarak assemblerin kabul edeceği komutlar dizisini girmektir. Programcı editörde yazdığı programı önce assembler butonuna basarak assembly diline uygunluğunu kontrol eder. Programı yapı bakımından oluşabilecek hatalardan arındırmak için hata ayıklama araçları tasarlanmıştır. Bu araçların yapabildikleri işlemler aşağıda sunulmaktadır:

- Programda kritik satırlarda durma noktası koyabilme,
- Seri çalışma dışında adımlayarak çalışma,
- Çalışmayı durdurma,
- Kaydedici, bellek ve PIA içeriklerini sıfırlayarak programı yeniden başlatma,
- İstenilen noktada kaydedici, bellek ve PIA içeriklerinin dökümünü görebilme,
- Hata sezmesi durumunda kullanıcıya programa kalındığı yerden devam edebilme,
- Kaydedicilerin içeriklerini inceleme ve değiştirme,
- Bellek içeriklerini inceleme,
- PIA portlarının içeriklerini inceleme imkanları vermesi şeklinde sıralanabilir.



Derlenmiş programın çalışması anında, kaydedici ve bellek içeriklerinin gösterimi hızlı olmasından dolayı kullanıcı tarafından takibi zor olmaktadır. Bu nedenle, simülatöre hata ayıklamada kullanılmak için işlevleri aşağıda verilmiş dört buton ve bir satır işleme hızını ayarlama çubuğu eklenmiştir (Şekil 5.4).



Şekil 5.4 Program çalıştırma seçenekleri

- Hız çubuğu ile seri çalıştırma modunda iken bir komut satırı çalıştırıldıktan sonra sıradaki komut satırını işleme almadan önce beklenecek süreyi ayarlamaktadır. Hız çubuğu 0-5000msn arasında 250msn aralıklarla ayarlanabilmektedir. Hız çubuğunda yapılan ayar animasyon penceresine etki etmemektedir. Animasyon penceresindeki iç çalışma kendi sabit hızında otomatik olarak ya da saat pulsü butonuna basılarak adım-adım yürütülmektedir.
- Yeşil trafik lambalı buton programı kaldığı yerden seri olarak çalıştırmaktadır. Durdurma butonuna basıldığından yada işletilen komut satırında durma noktası olduğundan program çalışması durmuşsa bu butona basılarak seri çalıştırılma başlatılabilir.
- Kırmızı trafik lambalı butona basıldığında programın seri çalıştırılması esnasında istediğimiz bir noktada durdurmak için kullanılır. Durma noktalarını kullanıcı önceden koymasına karşılık burada durdurma seçimi o anlık gerçekleşir.
- Adımlı buton ilk basıldığında adımlayarak çalışmayı başlatır ve her basıldığında sıradaki komut satırını çalıştırır.
- Reset butonu kaydedici, bellek ve PIA port içeriklerini sıfırlar ve programın çalıştırıcı sırasını ilk başlangıca yönlendirir.

Araç çubuğunun sağ tarafında o anda işlenen komut, kaydedici, bayraklar ve PIA içerikleri gösterilmektedir. Bu kısımdan kaydedici ve bayrak içeriklerinin son durumları değiştirilebilir. Bunun faydaları şunlardır:

- Sıradaki komutların çalışması kaydedici içeriklerinin farklı olduğu kombinezonlarla denenerek programın geliştirilmesine katkıda bulunmaktadır.
- Hali hazırda çalıştırılan komutlar istenilen kaydedici içeriklerini oluşturamamışsa, içerikleri değiştirebilme imkanı ile sıradaki komutların doğru çalışıp çalışmadıkları test edilebilmektedir.

Hata ayıklama aracı olarak yukarıda verilen çalıştırma butonları dışında içerikleri inceleyebilme ve değiştirebilme için işlem kütüğü, programın derlenmiş ve bellek penceresi mevcuttur.

İşlem kütüğü penceresi, her komut saykılında, mikroişlemcide icra edilen komut satırını, mikroişlemcinin kaydedicilerinde, PIA 8255 kaydedicilerinde ve bellekte yaptığı değişiklikleri incelemek amaçlı bir ekrandır (Şekil 5.5). En üst satırında o anda icra edilmekte olan komutun gerçekleştirdiği değişiklikleri gösterirken geriye yönelik olarak tüm işlenmiş satırların gerçekleştirdiği değişiklikleri hemen altında göstermektedir.

İşlem Kütüğü														Bellekteki İşlemler											
İşNo	PC	Komut	A	B	C	D	E	H	L	SP	S	Z	A	P	C	D	A	C <sub>H</sub>	B	C <sub>L</sub>	KÇ	Adr	O/Y	Veri	
6	0200	MVI A 20	20	00	00	00	00	00	00	00	00	00	00	00	00										
1	0200	MVI A 20	20	00	00	00	00	00	00	00	00	00	00	00	00	FF	FF	F	FF	F					
2	0202	MVI C 3E	20	00	3E	00	00	00	00	00	00	00	00	00	00	FF	FF	F	FF	F					
3	0204	MOV A B	00	00	3E	00	00	00	00	00	00	00	00	00	00	FF	FF	F	FF	F					
4	0205	MOV C D	00	00	00	00	00	00	00	00	00	00	00	00	00	FF	FF	F	FF	F					
5	0206	JMP 0200	00	00	00	00	00	00	00	00	00	00	00	00	00	FF	FF	F	FF	F					
6	0200	MVI A 20	20	00	00	00	00	00	00	00	00	00	00	00	00	FF	FF	F	FF	F					
																FF	FF	F	FF	F					
																FF	FF	F	FF	F					
																FF	FF	F	FF	F					
																FF	FF	F	FF	F					
																FF	FF	F	FF	F					

Şekil 5.5 İşlem kütüğü penceresi

İşlem kütüğü penceresinde: Komutların bellek okuma ve bellek yazma işlemleri adresleyen, adres, oku/yaz işlem türü ve veri olarak kaydedilmektedir. Adresleyenler PC, BC, DE, HL, SP kaydedici çiftlerinden biri olmaktadır. Pencerede işno üzerine fare ile tıklandığında “Bellekteki İşlemler” başlıklı kısımda işlenen komutun bellekte yaptığı tüm değişiklikler satırlar halinde gösterilmektedir.

İşlem kütüğünde, işlenen komutun içeriğini değiştirdiği kaydedici ve bayraklar renklendirilerek komutun etkilediklerinin fark edilmesi sağlanmaktadır.

Programın derlenmiş penceresinde, derleme işlemi sonucu hatasız bulunan satırlar, toplu halde gösterilmektedir. Adres kısmında belirlenen adrese tek tıkladığında adres kırmızı ve ikinci bir tıklamada adres yeşil hale gelmektedir. Kırmızı haldeki adres durma noktasıdır ve çalıştırma sırası o adrese geldiğinde çalışma otomatik olarak durmaktadır. Programın çalışması, “adım” ya da “çalıştır” butonuna basılarak devam ettirilmektedir.

Komut satırlarının bir altprograma daldığı veya G/Ç portlarından giriş ya da çıkış yaptığı önemli noktalarına durma noktaları konularak buralardaki yazılım-donanım ilişkisi ve kaydedici içerikleri kayıt altına alınabilir [14].

Bellek penceresi, program komutlarının, verilerin ve yığının olduğu bellek bölgelerinin birbirinden bağımsız olarak görülebilmesini sağlamaktadır. Gerçekleştirilen animasyonlarda tüm bellek dizisine bakılabilmektedir. İstenilen bir bellek dizisine hızla bakılabilmesi için, adresin yazılması ve git butonuna basılması yeterli olmaktadır. Aşağıdaki şekilde 0200 adresli program belleğinde derlenmiş bir programın “hex” kodları görülmektedir (Şekil 5.6).

Hata ayıklama işlemleri, mikroişlemci simülöründe grafiksel ortamda çok rahat bir şekilde yapılmaktadır.

Program	Veri	Yığın
0200 3E	0000 FF	FFF6 00
0201 20	0000 FF	FFF7 00
0202 0E	0000 FF	FFF8 00
0203 3E	0000 FF	FFF9 00
0204 78	0000 FF	FFFA 00
0205 4A	0000 FF	FFFB 00
0206 C3	0000 FF	FFFC 00
0207 00	0000 FF	FFFD 00
0208 02	0000 FF	FFFE 00
0209 00	0000 FF	FFFF 00

Şekil 5.6 Program, veri ve yığın bellek içeriklerini gösteren bellek penceresi

#### 5.4. Animatör

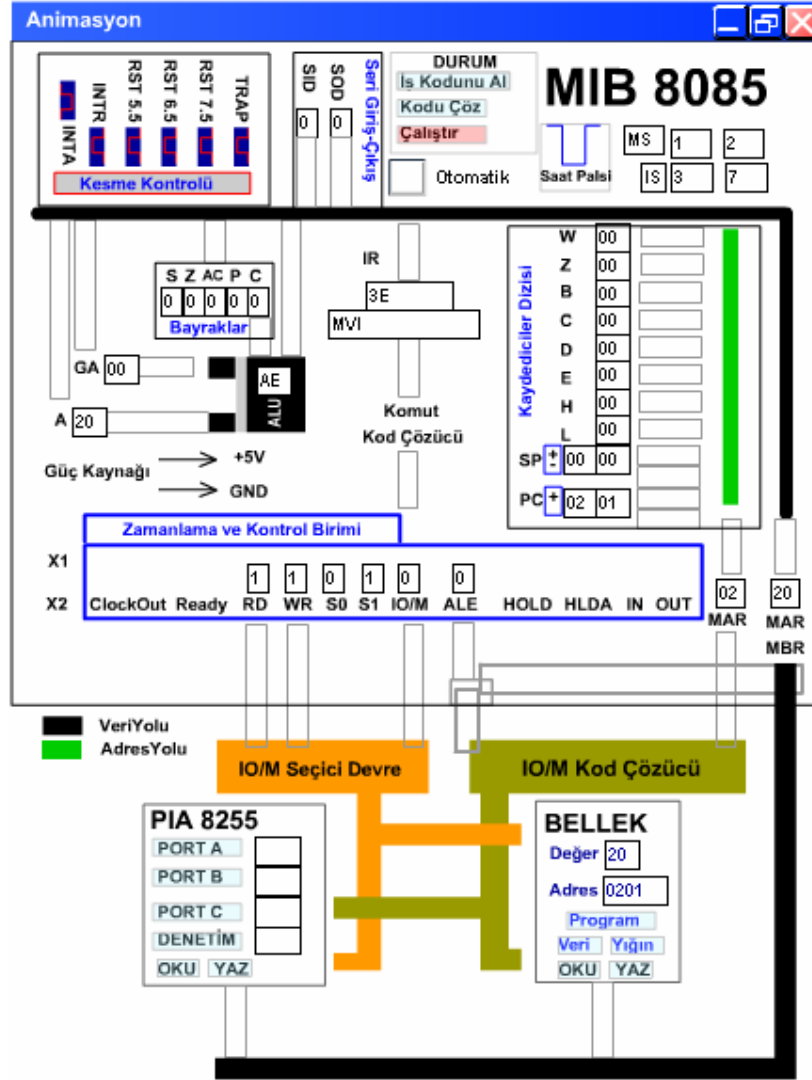
Bir mikroişlemci simülatörü en basit şekilde editör, assembler ve debugger'dan meydana gelmektedir. Bunlar, yazılan programa göre mikroişlemcinin yapması gereken işlemin gerçekleştirilmesinde gerekli olan unsurlardır. Kullanıcı tarafından assembly dilinde editörde yazılan program assemblerde makine diline çevrildikten sonra hatalardan arındırılması veya optimize edilmesi amacıyla debug işleminden geçirilir. Bu işlemler sonucunda programın nasıl çalıştığı ve ana birimler (kaydedici, ALU ve bellek) üzerindeki etkileri görülür. Fakat bütün bu işlemler sırasında veri yollarının durumu, kullanıcıya gözükmeyen fakat gerçekte işlemci içerisinde önemli görevler yüklenen bazı kaydedicilerin durumu, zamanlama ve kontrol biriminin etkisi, en önemlisi çalıştırılan program eğer bir G/Ç cihazı üzerinde denetim yapıyorsa o andaki etkilerinin görülmesi açısından simülatör tek başına yeterli olmayabilir [14]. Bu düşünceden yola çıkılarak simülatöre Şekil 5.7'de görülen, mikroişlemci, bellek, G/Ç birimleri ve iletişim yolları üzerindeki veri hareketlerinin gözlenebileceği bir animatör eklenmiştir.

Geliştirilen animatör, 8085 mikroşlemcili bir sistemde mikroşlemcinin kullandığı tüm komutların mikroşlemci mimarisinin içerisinde çalışmasını grafiksel olarak göstermektedir. Mikroşlemcili sistem; mikroşlemcinin kesme girişleri, seri giriş-çıkış uçları, programcıya görünen/görünmeyen kaydediciler, zaman-kontrol birimi, veri yolu, adres yolu, PIA ile bir bellekten oluşmaktadır.

8085 Mikroşlemcisi komut setinde 74 komut bulunmaktadır. Bu komutlardan her birisi farklı bir işlemi gerçekleştirir. Bir komutun icrası/işlenmesi sırasında mikroşlemci içerisinde gerçekleştirilen en küçük işlem zamanı "sistem saykılı" olarak isimlendirilir. Sistem saykılı, sistemin çalışma hızını belirten tetikleme sinyali frekansı ile (sistem saat saykılı) belirlenir.

Mikroşlemcili sistemde, komutların çalışması sistem saykılı seviyesinde en küçük harekete kadar canlandırılmaktadır. Sistemde tüm komutlar canlandırıldığı için, kullanıcının sistemin iç çalışması ile ilgili eksik kalacağı bir nokta kalmamaktadır. Mikroşlemcili sistemde, her komutun kullandığı adresleme modu ve yaptığı işlem

farklıdır. Kullanıcı yapılan canlandırmalar ile tüm mikroişlemcilerde benzer yapıda olan ve kavranması güç olan adresleme modları hakkında bilgi sahibi olmaktadır.



Şekil 5.7 Mikroişlemci mimarisi ve bellek elemanlarını içeren animasyon penceresi

8085 mikroişlemcisinde; ivedi adresleme, doğrudan adresleme, kaydedici adresleme, kaydedici dolaylı adresleme ve bellek dolaylı adresleme olmak üzere beş farklı adresleme yöntemi kullanılmaktadır [22]. Komutların çalışmasının canlandırılması aşamasında, adresleme modlarının mikroişlemcinin hangi elemanlarını hangi sırada kullandığı gösterilmektedir.

Canlandırmalar, her komut için makine saykılı ve daha alt birim olarak sistem saykılı

bazında yapılmaktadır. “Saat palsi” butonuna her basıldığında, işlenmekte olan komutun sıradaki sistem saykılında yapılması gereken işlemler canlandırılmaktadır. Kullanıcı otomatik seçeneğini seçerek bir komut saykılı boyunca mikroişlemcili sistemde neler gerçekleştiğini “saat palsi” butonuna basarak adımlamak yerine komut saykılı bitinceye kadar adımların ardışık işlenmesini izleyebilmektedir.

Animasyon pencerenin sağ üstünde işlenecek komutun kaçınıcı makine ve sistem saykılında bulunduğu yanlarındaki kutularda ise toplamda kaç makine ve sistem saykılı tutacağı görülmektedir (Şekil 5.7).

Canlandırma esnasında pencerenin üst kısmında durum adı altında gruplanan “İş Kodunu Al”, “Kodu Çöz” ve “Çalıştır” durumları adımlama sırasında gerçekleştiklerinde zemin rengi kırmızı olarak renklendirilmektedir. Aynı şekilde bellek adı altında gruplanan “OKU”, “YAZ” seçenekleri ve PIA 8255 adı altında gruplanan “PORTA”, “PORTB”, “PORTC”, “DENETİM”, “OKU”, “YAZ” durumları adımlama sırasında gerçekleştiklerinde zemin renkleri kırmızı olarak renklendirilmektedir. Böylece kullanıcının mikroişlemci içerisinde komutun icrasında gerçekleşen işlemleri kolay anlaması sağlanmaktadır.

Tüm komutlarda öncelikle PC kaydedicisinin gösterdiği bellek adresinden iş kodu okunmaktadır. Daha sonra IR komut kod çözücünün belirlemesiyle işletilmekte olan komutun kullandığı adresleme moduna göre bellek oku - bellek yaz alt işlem parçalarından uygun olanlar ardışık olarak gerçekleşmektedir. Komutların işlenmesinde ortak olan bu alt işlem parçalarının her birisine “makine saykılı” denir. Komutların işlenmesi sırasında aşağıda verilen beş temel makine saykılı oluşur:

1. İş kodu okuma
2. Bellek okuma
3. Belleğe yazma
4. G/Ç terminali okuma
5. G/Ç terminali yazma

Bir makina saykılı sırasında gerçekleştirilen her hangi bir işlem, birkaç aşamada gerçekleştirilir. İşlemin özelliğine göre, gerçekleştirilme aşamalarının sayısı değişir.

İşlemin aşamalarının sayısının, komutun makina saykılıının anlaşılmasına yardımcı olması amacıyla; işkodu alma, bellek okuma ve bellek yazma işlemlerinin makine saykılı ile ilişkisini inceleyelim.

#### 5.4.1. İş kodu okuma

Herhangi bir komutun işlenmesinde ilk işlem, işkodunun alınmasıdır. Bir komutun işlenmesine başlamadan önce, işkodunu temsil eden bilginin bulunduğu bellek bölgesinden alınması gerekir. İşkodunun alınması işleminin, adres/veri yolu ve kontrol sinyali ile ilişkisi Şekil 5.8’de blok şema olarak gösterilmektedir. İşkodunun alınması işleminin zamanlama diyagramı ise Şekil 5.9’da verilmektedir.

MOV C, A komutu ile (4F) oluşan olayları inceleyelim. İşkodu alma işleminde;

İşkodunun bulunduğu bellek bölgesinin adresi, PC tarafından adres yoluna yerleştirilir. Komutun işkodunu temsil eden makine kodu (4FH) bulunduğu yerden okunur (Şekil 5.9). Okuma işlemine, kontrol birimi tarafından gönderilen ‘RD’ sinyali ile yetki verilir. Okunan bilginin işkodu olduğu, durum sinyallerinin ((IO/M) S0 ve S1) değerleri tarafından belirtilir.  $IO/\overline{M} = 0$  olması durumu; işlemin bellek ile ilgili olduğunu ve S1=S0=1 olması durumu; işkodu alma işlemi olduğunu belirtir. Alınan işkodu, komut kod çözücü tarafından çözümlenerek yapılacak işlemler belirlenir.

8085 mikroişlemcisinde kullanılan komutların bir kısmı üç sistem saykılında işlenirken, bir kısmı dört veya altı sistem saykılında işlenir. Şekil 5.9’da açıklanan işlem aşamaları sırasında, sistem saykılı (saat sinyali) ile ilişkili beş farklı sinyalin durumları gösterilmektedir. Açıklanan sistem saykılıları sırasında, adres ve veri yolu paralel hatlar olarak gösterilmektedir. Bunun nedeni; adres ve veri yolunda bulunan hatların bir kısmı lojik ‘1’ olurken, diğer bir kısmının lojik ‘0’ değerine sahip olmasıdır. Bu hatların yeni değerler alması durumu, çapraz çizgiler ile belirtilirken, hatların yüksek empedansa sahip olması durumu kesik çizgiler ile belirtilir.

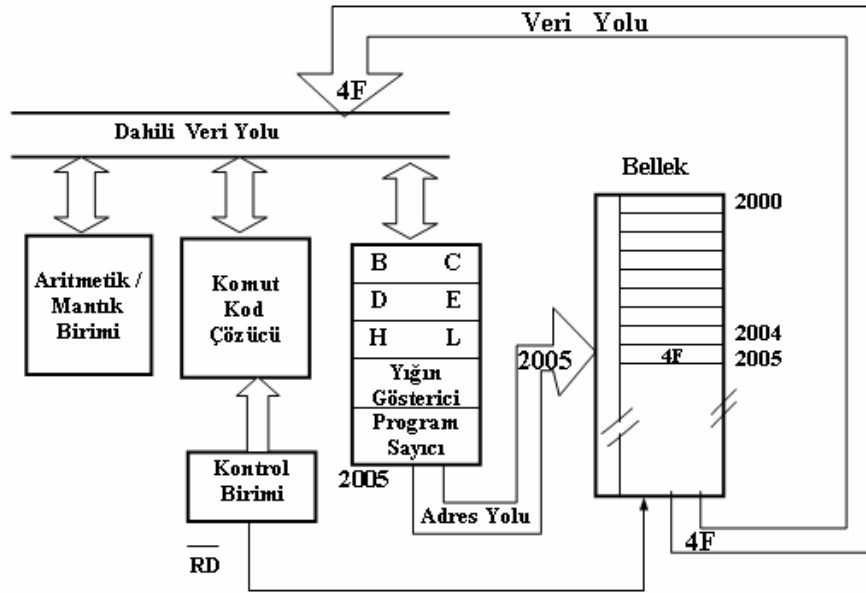
İşkodu alma işleminin T<sub>1</sub> sistem saykılı sırasında bellek adresinin yüksek değerli

kısmı (20H)  $AD_8-AD_{15}$  nolu adres hatlarına yerleştirilirken, bellek adresinin düşük değerli kısmı  $AD_0-AD_7$  nolu adres hatlarına yerleştirilir. ALE sinyali; lojik '1' değerini alırken ( $A_0-A_7$  hatlarının adres hattı olduğunu belirtir),  $IO/\overline{M}$  sinyali, işlemin bellekle ilişkili bir işlem olduğunu belirtmek için lojik '0' değerini alır.

$T_2$  sistem saykılı sırasında  $\overline{RD}$  kontrol sinyali lojik '0' değerine sahiptir ve bu sinyal bellek entegresini yetkilendirir. Bellek entegresinin yetkilenmesi ile komut kodu (4F)  $AD_0-AD_7$  nolu adres hatlarına yerleştirilir ve mikroişlemciye aktarılır. Diğer bir deyişle;  $\overline{RD}=0$  sinyali, 4F değerinin veri yoluna ( $AD_0-AD_7$ ) yerleştirilmesini sağlar.

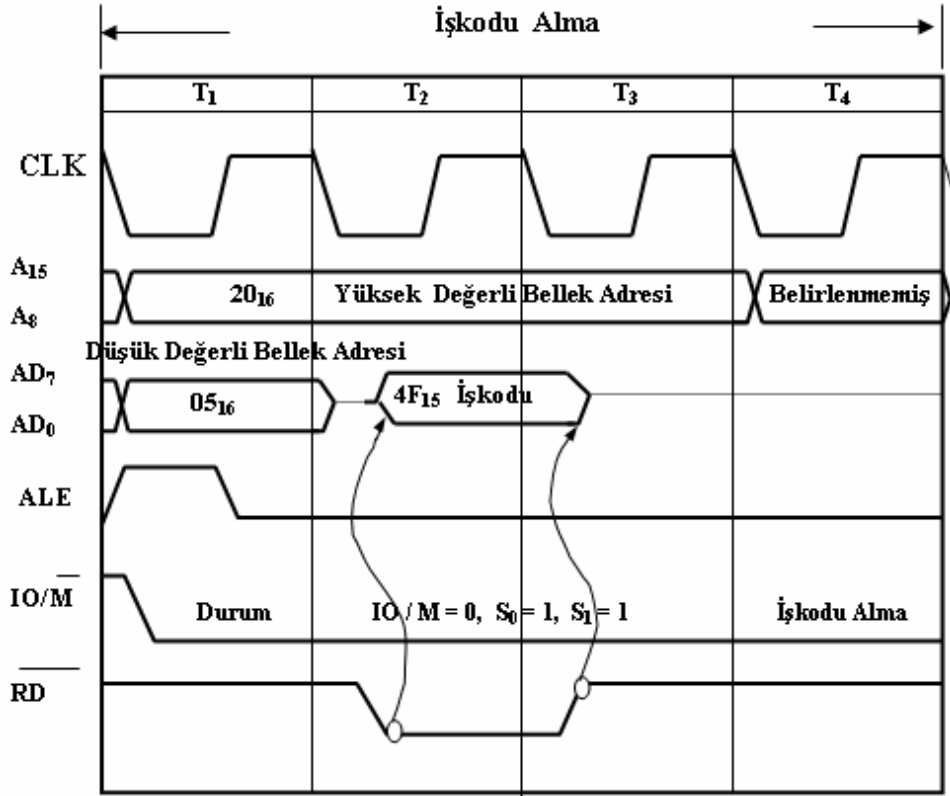
$\overline{RD}=1$  olduğu  $T_3$  sistem saykılı anında, veri yolu yüksek empedans durumuna geçer.

$T_4$  sistem saykılı sırasında,  $(4F)_{16}$  makine kodu komut çözücü tarafından çözülür ve akümülatörün içeriği C kaydedicisine kopyalanır.  $T_4$  anında, işkodu alma işlemi bitirilir.



Şekil 5.8 İşkodu alma işleminde oluşan olaylar





Şekil 5.9 Bellekten mikroişlemciye bilgi aktarımı işleminin zaman diyagramı.

#### 5.4.2. Bellek okuma ve bellek yazma makine saykılıarı

Bellek okuma makine saykılıarını inceleyebilmek için, 2 veya 3 bayt'lık komutları incelememiz gerekir.

Örnek olarak; 'MVI A, 32H' komutunun işlenişini inceleyelim. Komut ile 32<sub>16</sub> bilgisinin akümülatöre yüklenmesi hedeflenmektedir.

İşlenecek komut 8085 mikroişlemcisinde (3E 32)<sub>16</sub> sayısı ile temsil edilir. Bu iki baytık komut bilgisini 2000H-2001H adresli bellek bölgelerinde yüklü kabul edelim. İlk makine kodu (3EH) iş kodunu temsil etmekte ve akümülatöre bir bayt'lık veri yüklenmesi işlemini, ikinci makine kodu ise (32H) akümülatöre yüklenecek değeri temsil etmektedir. Bu makine kodlarının işlenmesi sırasında gerçekleştirilen işlem aşamalarını ve makine saykılıarında oluşan olayları detaylandıralım.

Komutta; birinci bayt işkodu, ikinci bayt veri olmak üzere 2 bayt bulunmaktadır.

Mikroişlemcinin bu bilgileri bellekten okuması en az iki makine saykılı gerektirir: İşkodu alma ve bellek okuma. İki makine saykılı, toplam 7 sistem saykılı içermektedir.

İlk makine saykılı sırasında oluşan olaylar, yollardaki değerler haricinde Şekil 5.9’da açıklanan işlemlerin aynısıdır. İşkodu alma işleminin tamamlanması ile 2001H değeri adres yoluna yerleştirilir ve durum sinyallerinin değerleri ile ( $IO/M=0$ ,  $S_1=0$ ,  $S_0=1$ ) bellek okuma saykılı olduğu bulunarak  $ALE = 1$  değerine kurulur.  $T_2$  sistem saykılı sırasında, RD sinyali ile bellek entegresi aktif yapılır ve  $T_2$  sinyalinin yükselen kenarında 32H bilgisi veri yoluna yerleştirilir.  $T_3$  anında, veri yolundaki bilgi akümülatöre yüklenir.

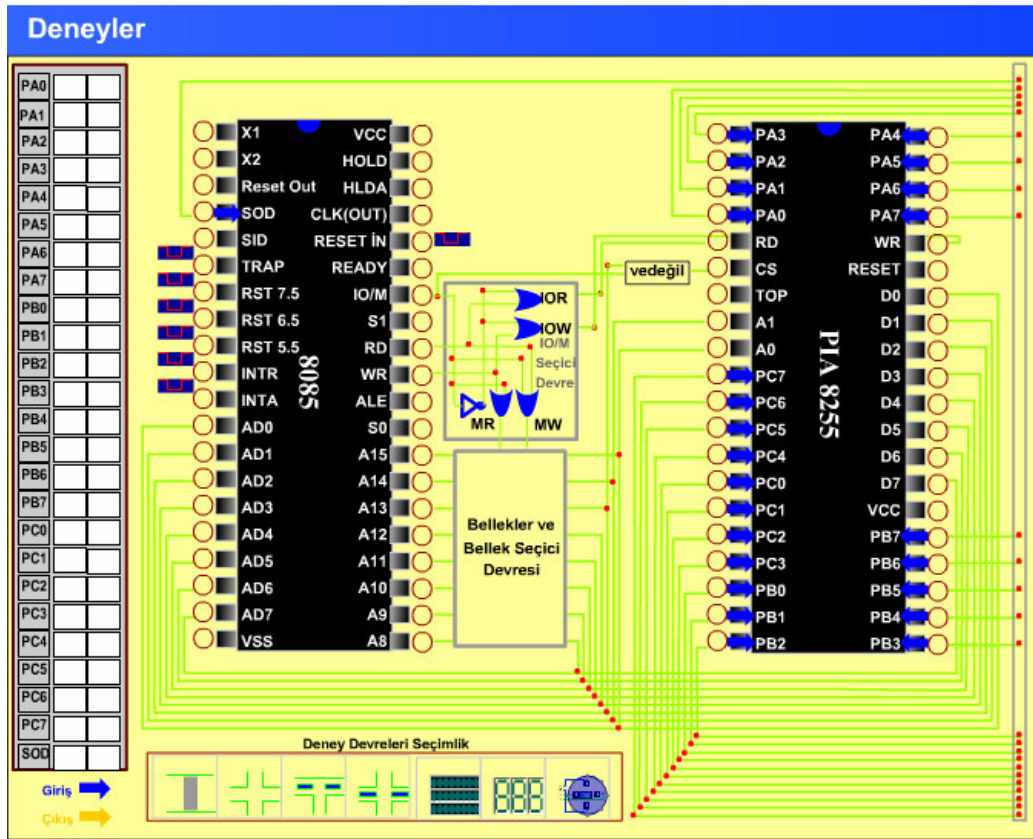
Komutların açıklanması için kullanılan bayt sayısı ile komutu işlemek için gerekli makine saykılı arasında doğrudan bir ilişki yoktur. Bu durumu açıklamak için; ‘STA 2065H’ komutunu işlenişini inceleyelim: Komut ile akümülatörün içeriği,  $(2065)_{16}$  adresli bellek bölgesine yazılmaktadır.

İncelenen komut 3 bayt’lık bir komut olmasına rağmen, 13 sistem saykılı içeren 4 makine saykılına sahiptir:

1. “2010H” adres değeri adres yoluna yerleştirilir ve bildirilen adreste bulunan ‘32H’ işkodu bilgisi alınır.
2. ‘2011H’ adres değerini adres yoluna yerleştirir ve bu bellek bölgesinde bulunan ‘65H’ düşük değerli bellek bölgesi adres değerini okur.
3. ‘2012H’ adres değerini adres yoluna yerleştirir ve bu bellek bölgesinde bulunan ‘20H’ yüksek değerli bellek bölgesi adres değerini okur.
4. Son makine saykılı, bellek yazma işlemidir. 2065H adres bilgisi, adres yoluna yerleştirilir ve durum sinyallerinin değeri ile ‘bellek yazma’ işlemi olduğu anlaşılır ( $IO/M=0$ ,  $S_1=0$ ,  $S_0=1$ ). Akümülatörün içeriği, veri yoluna ( $AD_7-AD_0$ ) yerleştirilir ve  $\overline{WR}$  sinyali üretilir. Üç sistem saykılı kullanılan bellek yazma işleminde, son sistem saykılı sırasında veri yolundaki bilgi ‘2065’ adresli bellek bölgesine yerleştirilir.

## 5.5. Deneysel Devreleri ve Kullanımı

Kullanıcı tarafından assembly dilinde yazılan bir program derleme işlemine tabi tutulur. Kullanıcı, derlenmiş programın çalıştırılması sonucu işlem kütüğü ve bellek pencerelerinde kaydedici, ALU, bellek ve PIA portları üzerindeki etkilerini görmektedir. Animatörde ise derlenmiş programın çalıştırılması sonucu veri yollarının durumu, kullanıcıya gözükmeyen fakat gerçekte işlemci içerisinde önemli görevler yüklenen bazı kaydedicilerin durumu, zamanlama ve kontrol biriminin etkisi ve G/Ç cihazı üzerindeki denetim etkileri görülmektedir. Ancak tüm bunlar günlük hayatta mikroişlemcinin hangi çevre cihazlarıyla ne gibi denetimler yapmakta olduğunu görebilme ve deneme yapabilmeye imkanı vermez. Bu düşünceden yola çıkarak simülatöre, mikroişlemci, bellek, PIA ve çevre birimlerinin örnek uygulamalarından oluşan deneyler penceresi eklenmiştir (Şekil 5.10).



Şekil 5.10 Mikroişlemci deney uygulamaları penceresi

Deneyler penceresinde PIA 8255 entegresinde PA0-PA7, PB0-PB7 ve PC0-PC7

ayakları ve 8085 entegresinde SOD ayağına bağlantı kurulabilmektedir. Bağlantı kurulabilen bu uçlara komut satırının işlenmesi sonucu ya da bağlı anahtarlar dolayısıyla gelen değer lojik “1” ise entegre ayağının yanındaki daire kırmızı renk yanmakta aksi durumda içi boş görünmektedir. Port ucu giriş yönünde kurulu ise ucun üzerinde bulunan ok mavi ve yönü port girişine doğru, çıkış yönünde kurulu ise ok portakal renkli ve yönü port ucuna ters duruma getirilerek kullanıcının port uçlarının giriş/çıkış yönünde kurulu olduğunu anlaması sağlanmaktadır.

Deneyler penceresinde PIA 8255 ile çalışmayı göstermek amacıyla sadece mod0 modunda kullanılmasına izin verilmekte, mod1 ile mod2 için kullanıcıya uyarı verilmektedir. PIA 8255 entegresini seçme işlemi için 8085 IO/M ve AD2 pinleri “VEDEĞİL” lojik kapısıyla PIA 8255 CS girişine verilmiştir. Bundan dolayı “IN” ve “OUT” komutları ile üzerinde işlem yapılacak port seçim değerlerinin 2.bit değeri “1” olmalıdır.

Kullanıcı istediği PIA ayağına deney yapmak istediği deney modülündeki elemanın ayağını bağlayabilir. Bunun için PIA ayağına fare ile tıkladıktan sonra deney modüllerinde bulunan trafik lambalarından istediğini, 8-bitlik LED’lerden istediği LED’i, yedi parçalı göstergeyi veya adım motorunu tıklamak suretiyle bağlantı kurabilir. Bağlantılar PIA portlarından düzenlenmiş hat üzerinden bağlantı kurulacak elemana doğru diğerlerinden farklı renkte çizgi ile gösterildiği gibi deneyler penceresinin sol kısmındaki listede de gösterilmektedir.

IN komutu için PIA portlarının alacağı değer deneyler penceresinde anahtarların PIA port uçlarına bağlantısı ve konumlarına (açık/kapalı) göre değişmektedir. Animasyon penceresi ve araç çubuğu kısmından port değerlerinin değiştirilmesine izin verilmeyerek pencereler arası uyumluluk sağlanmaktadır.

Deneyler penceresinin sağ üst kısmında uygulama modülü seçme işleminin yapıldığı seçimlik kısmı Şekil 5.11’de görülmektedir. Simülâtör programlarının en büyük artısı sayılabilecek özelliklerinden birisi; yeni modüllerin eklenmesinin kolay olmasıdır. Tasarlanan deneyler içerisinde seçilebilecek 7 uygulama parçası bulunmaktadır.



Şekil 5.11 Uygulama modülü seçim penceresi

Uygulama parçaları, dört adedi trafik ışık kontrolü, bir adet (kendi içinde 3 adet) 8 bitlik LED grubu, 1 adet (kendi içinde 3 adet) yedi parçalı gösterge ve 1 adet adım motorundan oluşmaktadır (Şekil 5.11). Adım motoru dışındaki tüm elemanlar gerçekte LED mantıklı elemanlardır.

### 5.5.1. Trafik ışık kontrolü uygulamaları

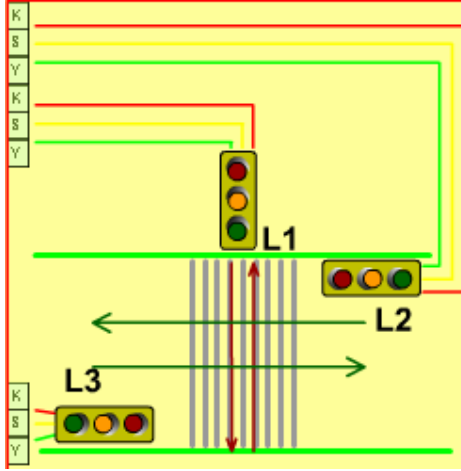
Sinyal olarak adlandırılan ışıklı işaretler; yollarda ve özellikle kavşaklarda düzenli ve güvenli bir trafik akışı sağlamak için kullanılan kontrol gereçleridir [33].

Tasarlanan uygulamalar ile belirli özelliklere sahip kavşakların trafik akışı sağlanarak, ışık sisteminin mikroişlemci ile kontrolü açıklanıp, belirli kavşaklardaki uygulamaları gösterilir. Böylece kullanıcının günlük hayattan örnekler üzerinde çalışması sağlanır.

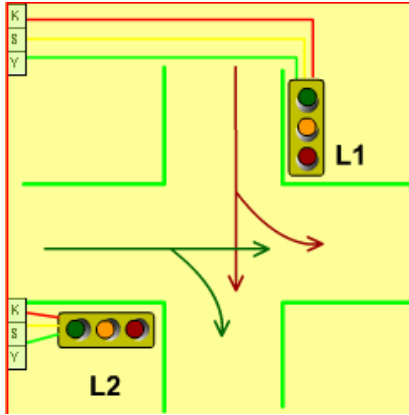
Trafik ışık kontrolü uygulamaları, kullanıcıya trafik lambalarındaki kırmızı, sarı ve yeşil ışıkların yanış sürelerinin programlanmasını öğretmek içindir. Trafik lambaları, gecikme amaçlı programların geliştirilmesinde en etkin araçtır.

Bir sinyal devresi içerisinde bir veya birden fazla trafik akımını aynı anda öngören kumanda şekline “Faz Yöntemi” adı verilir. Sinyalizasyon sisteminde seçilecek faz yöntemi, kavşağa girişi olan yol sayısına ve kesişen trafik yoğunluğuna bağlıdır. Bu değerlere bağlı olarak 2, 3, 4 fazlı sistemler kullanılır. Faz sayısının az olması kayıp zamanı azaltacağından mümkün olduğu kadar az faz kullanılmalıdır [33].

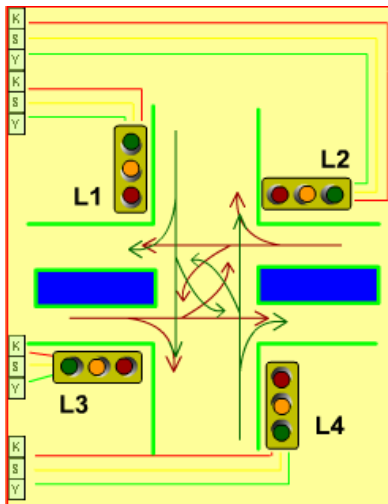
Trafik ışık kontrol modülleri sırasıyla “Yaya geçidi” Şekil 5.12’de, “Tek yönlü yaya kavşağı” Şekil 5.13’de, “Sola dönüşün az olduğu kavşak” Şekil 5.14’de ve “Tek yönlü T kavşak” uygulaması Şekil 5.15’de gösterilmektedir.



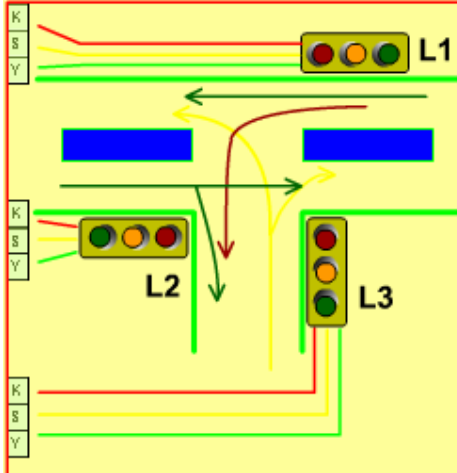
Şekil 5.12 Yaya geçidi



Şekil 5.13 Tek yönlü yaya kavşağı



Şekil 5.14 Sola dönüşün az olduğu kavşak



Şekil 5.15 Tek yönlü T kavşak

Kullanıcının fazları ayırabilmesi için uygulama ekranlarında farklı renkte yön çizgileri kullanılmıştır. Trafik lambalarından çıkarılan uçlar PIA portlarından düzenlenmiş hatta yanaşık düzende yerleştirilmiştir.

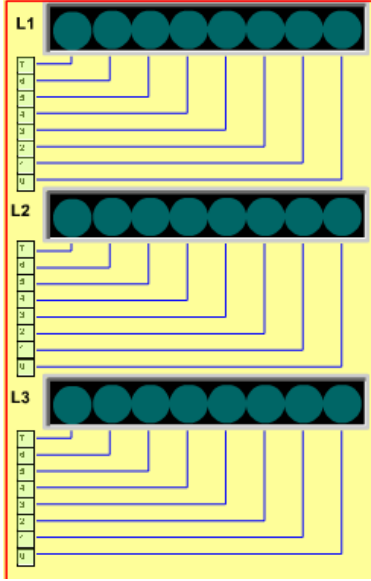
Kullanıcı bağlantı kurmak istediği PIA ayağına fare ile tıkladıktan sonra trafik lambalarından istediğinin herhangi renkteki lambasına yada lambalardan çıkartılarak düz bir hat şeklinde dizilmiş uçlara tıklayarak bağlantı kurabilmektedir.

### 5.5.2. 8-bitlik led paneli deney modülü

Gerek elektromekanik gerekse simülatör temelli öğretim araçlarında en kullanışlı kontrol elemanı LED'lerdir. Görsel, ucuz ve küçük ebatlı olmalarından dolayı elektronik sistemlerde durum işaretçisi olarak sık kullanılmaktadır [14].

Kullanıcı bağlantı kurmak istediği PIA ayağına fare ile tıkladıktan sonra istediği bir led'e yada led'den bir bağlantı ile çıkartılarak düz bir hat üzerine konulmuş uca tıklayarak bağlantı kurabilmektedir (Şekil 5.16).

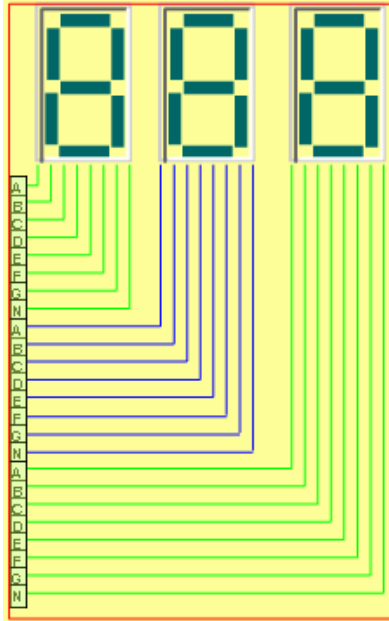
Uygulamalarda, dizi şeklindeki LED'ler üzerinde birçok sanal işlem gerçekleştirilebilir. Bunlar, LED'lerin sırasıyla sağa ve sola doğru yakılması, ileri veya geri saydırılması olabilir.



Şekil 5.16 Üç adet 8-Bitlik LED paneli

### 5.5.3. Yedi parçalı gösterge uygulaması

Elektronik sistemlerde veri akışının basit yöntemle gözlenmesini sağlayan yedi parçalı göstergenin her bir parçası yedi adet LED'ten meydana geldiğinden üzerinde basit bilgiler gösterilebilmektedir (Şekil 5.17). Bu kolaylıktan faydalanarak, gösterge üzerine çeşitli harf veya sayılar yazdırılarak kaydırılabilmektedir [14].



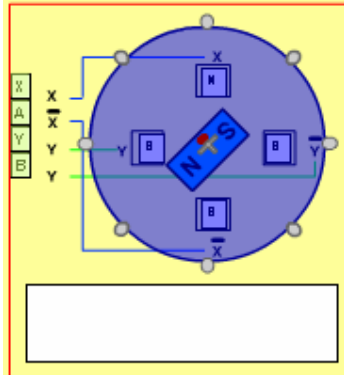
Şekil 5.17 Yedi parçalı gösterge uygulaması



#### 5.5.4. Adım motoru deney modülü

En basit şekliyle disket sürücülerin içerisinde bulunan adım motorları yaygın kullanıma sahip elektromekanik bir elemandır.

Adım motorların dönen kısmı (rotor) sabit mıknatıstan yapılmıştır. Duran kısmında (stator) ise belirli aralıklarla yerleştirilmiş elektromıknatıslar bulunmaktadır (Şekil 5.18). Elektromıknatısın içerisinde geçen akımın yönüne göre N-S kutuplarının yönü de değiştirilebilmektedir. Bir adım motorun döndürülmesi için belli bir sırayla bu elektromıknatısların enerjilenmesini sağlayan gerilimler motor uçlarından uygulanır. Böylece rotordaki sabit mıknatıs, statorun enerjilenen kutupları tarafından yönlendirilir (N-S kutupları birbirini çeker, N-N veya S-S kutupları birbirini iter) [19].



Şekil 5.18 Adım motoru ve iç yapısı

4 uçlu bir adım motorun sağa dönmesi için uçlarına Tablo5.1'de verilen gerilimlerin uygulanması gerekmektedir.

Tablo 5.1 Adım motoruna verilecek kutuplaşma gerilimleri

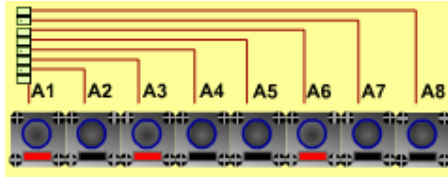
Adım	X	X	Y	Y
1	1	0	0	0
2	1	0	0	1
3	0	0	0	1
4	0	1	0	1
5	0	1	0	0
6	0	1	1	0
7	0	0	1	0
8	1	0	1	0
9(1)	1. adım ile aynı			

Tablo5.1’de verilen lojik “1” deęerleri motorun uçlarına uygulanacak olan pozitif gerilimleri ifade etmektedir.

### 5.5.5. Anahtarların giriş elemanları olarak kullanılması

Simülatörde çevre giriş elemanı olarak bir adet anahtar grubu kullanılmıştır (Şekil 5.19).

Anahtarlar üzerinde biri daire şeklinde, dięeri dikdörtgen şeklinde iki buton yer almaktadır. Daire şeklinde olanına basılı tuttuęunuz müddetçe çalışırken, dikdörtgen şeklinde olan butona tıklanarak anahtar basılı pozisyonda tutulmakta ya da tekrardan serbest bırakılmaktadır.



Şekil 5.19 Anahtar grubu

## BÖLÜM 6. PIC16F84 SİMÜLATÖRÜ

### 6.1. Genel Tasarım

PIC16F84 Simülatörü, assembly dilinde program yazılabilecek bir metin editörü (düzenleyici), yazılan bu programı derleyebilecek bir assembler, derlenen programın hatalarını bulacak ve ayıklayabilecek bir debugger, mikrodenetleyici içerisindeki olayların gözlemlendiği simülatör ve yazılan programlara uygun olarak çalışan uygulama bölümlerinden oluşmaktadır. Tüm bu bileşenler internet tarayıcı yada Flash player bulunan tüm işletim sistemlerinde çalışmaktadır.

Simülatör üzerinde gerçekleştirilen tüm işlemler hazır komutlar yerine ilgili butonlara fare ile tıklamak suretiyle yapılmaktadır.

Araç çubuğu penceresinin sol kısmında RAM, EEPROM, bellek, işlem kütüğü, komut editörü, programın derlenmiş, animasyon ve deneyler penceresini açma/kapama butonları bulunmaktadır (Şekil 6.1). Bu butonlara bir kere basıldığında pencere açılmakta ikinci basıldığında pencere kapatılmaktadır.

Araç çubuğunda pencere açma/kapama butonlarından sonra sırayla yazılan programı derleme ve çalıştırma ile ilgili butonlar bulunmaktadır.



Şekil 6.1 PIC16F84 simülatörü araç çubuğu

## 6.2. Editör Tasarımı

Simülatörde kullanılan metin düzenleyicisi (editör), PIC16F84 mikrodenetleyici için uygun komutlar kullanılarak assembly dilinde program yazılmasını sağlayan simülatörde yerleşik bir araçtır. Yazılan program araç çubuğu üzerinde bulunan assembler butonuna basılarak derlenebilir. Editör penceresi üzerinde işletim sisteminin desteklediği kes, kopyala ve yapıştır gibi düzenleme işlemleri klavye kısayolları yada farenin sağ tuş menüsü seçenekleriyle yapılabilmektedir. Programın sabit diske kaydedilmesi editörün yazıldığı Flash programının yetenekleri dahilinde olmadığından program simülatörden başka bir metin editörüne kopyalanarak kaydedilebilir.

Editörde program yazarken, programın derlenmesi aşamasında istenmeyen hatalar ortaya çıkmaması için assembly dilinin yazım kurallarına uyulması zorunludur. Editör penceresinde assembly dilinde program yazarken dikkat edilecek kurallar:

- Assembly dilinde yazılan her komut satırı sırayla etiket, iş-kodu (operatör-komut), işlenen (operand) ve açıklama işlem alanlarından oluşmaktadır.
- İşlem alanları birbirinden boşluk veya “,” karakteriyle ayrılmalıdır.
- Gereğinden fazla boşluk konulmamalıdır. Aksi takdirde programın okunabilirliğini azaltmaktadır.
- Komutlara ait parametreler birbirlerinden “,” karakteriyle ayrılmalıdır.
- İşlenen alanında komuta ait parametre değer ifade edecekse ilk karakteri sayı sistemini belirten H (hex-onaltılı), B (binary-ikili) veya D (decimal-onluk) harflerinden biri olmalıdır. Değer çift tırnak arasında yazılmalıdır. Örneğin; H"EF".
- Etiket olarak seçilen isimlerin ilk karakteri hiçbir zaman sayısal bir değer olamaz. İlk karakter harf seçildikten sonra diğerleri sayısal bir karakter olabilir.
- Etiket isimlerinde noktalama işaretleri gibi işaretler olamaz. Etiket ismi harf ve sayılardan oluşur.
- Aynı etiket iki kere kullanılamaz (derleyici ilk etiketi geçerli kabul edecektir) ancak birden fazla işlenen alanında kullanılabilir.

- Programın okunabilirliğini arttırmak için gerekli açıklamalar, komut satırının bitiminden başlar. Dikkat edilmesi gereken nokta, assemblerin bu açıklamaları göz önüne almamasını sağlayacak noktalı virgül (;) işaretinin hemen açıklamaların başına konulması gerekliliğidir.
- Kullanıcı atama deyimi EQU ile programda kullanmak üzere kendi sabitlerini tanımlayacaksa bunu program bloğundan önce yapmalıdır. Aksi durumda kullanıcının tanımladığı sabitler program bloğunda dikkate alınmayacak ve assembler işleminde o program bloğundan sonra tanımlanmış bu sabitleri içeren komut satırları derlenemeyecektir.
- Komut kodlarının belleğe yerleşeceği adres, komutlardan önce kullanılacak ORG komutu ile belirlenir. Örnek: ORG H"0200" satırından sonra gelecek komutlar belleğe H"0200" adresinden yerleşmeye başlayacaktır.

Metin editöründe komut satırı tamamen büyük harfe çevrilerek derlendiğinden etiket, komut yada işlenenlerde büyük yada küçük harf kullanılması fark etmemektedir.

Assembly dilinde satırda sadece etiket varsa tek etiket olan satırdan sonra gelen satır komut ile başlıyorsa etiket o satırı göstermektedir. Yazılan editörde satırdaki tek kelime etiket kabul edilmesine rağmen sonraki satıra ait olarak kabul edilmemektedir.

Yazılan editörde Assembly dilinde komutları yazılırken çağırma komutları ile birlikte etiket kullanılmaktadır. Programlama dillerinde etiketin kullanılma sebepleri şunlardır;

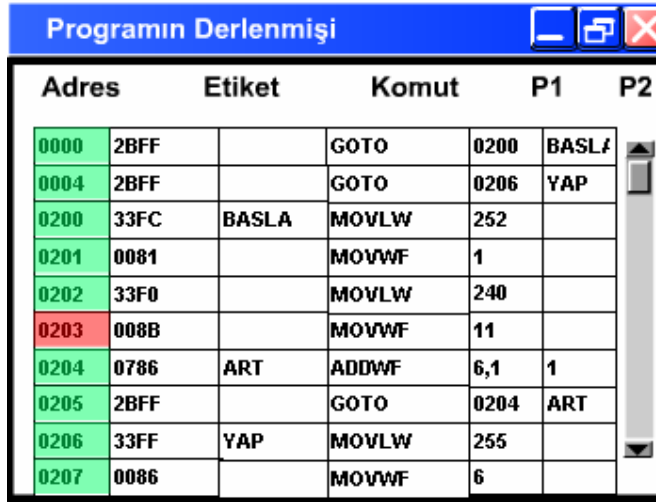
- Program alanının kolaylıklar bulunmasının ve hatırlanmasını sağlar,
- Programda bir değişiklik veya yeniden düzenleme yapıldığında, belirlenen adresler kayacaktır ancak etiketlenmiş blok önceki belirlenen aynı adresi korumuş olacaktır.

### 6.2.1. Derleyici tasarımı

Bu bölümde, simülatör içerisinde yer alan PIC16F84 assemblerinin ayrıntılı tasarım

ve yürütülme aşamaları ele alınmaktadır.

Assembler tarafından hex kodlar haline getirilen program, gerçek bir mikrodenetleyicide veya bir simülatörde çalıştırılabilir. Assembler işlemi sonunda elde edilen assembly kodlarla birlikte onaltılık düzende makine kodları ve bu kodlara karşılık gelen adreslerin yer aldığı Programın Derlenmiş penceresi Şekil 6.2’de gösterilmektedir.



Adres	Etiket	Komut	P1	P2	
0000	2BFF		GOTO	0200	BASLA
0004	2BFF		GOTO	0206	YAP
0200	33FC	BASLA	MOVLW	252	
0201	0081		MOVWF	1	
0202	33F0		MOVLW	240	
0203	008B		MOVWF	11	
0204	0786	ART	ADDWF	6,1	1
0205	2BFF		GOTO	0204	ART
0206	33FF	YAP	MOVLW	255	
0207	0086		MOVWF	6	

Şekil 6.2 Derleme işlemi sonuç listesi ve durma noktaları seçimi

Programın hex kod düzeninde işkodu haline getirilmesi için hedef mikrodenetleyicinin kullanılmasına gerek yoktur.

Derleme işleminde yapılanlar:

- Program chr(13)-Enter karakterine göre satırlara ayrıştırılmaktadır.
- Satırlar sırayla (“;” karakteri öncesi yani açıklamaya kadar olan kısımları) satır kontrol işlemine tabi tutulmaktadır.
- Satır kontrol işlemlerinde program bloğundan önce atama bloğundaki “EQU” etiketli satırlar kullanıcı sabiti olarak ayrı bir diziye kaydedilmektedir.
- Satır kontrol işleminde değerlendirilen satırlardan hatasız olanlar çalıştırılacaklar dizisinde, dallanma ve atlama komutlarının çağırdıkları etiketler ve buldukları

satırın numarası çağırın dizisinde, satırların etiketleri ise etiket dizisinde tutulmaktadır.

- Çağırın dizisinde tutulan etiketler sırayla etiket dizisiyle karşılaştırılmakta ve karşılığı olmayan etikete sahip satırlar çalıştırılacaklar dizisinden silinmektedir.
- Çalıştırılacaklar dizisinden silinmiş satırlar olacağı için etiketlerdeki satır numaraları tekrardan düzenlenmektedir.
- Çağırınlar için son kez etiketlere karşılık gelen hex kodları düzenlenmektedir.

Satır kontrolü işleminde yapılanlar:

- Satırı değerlendirirken, yazılan satırı boşluk ve “,” karakterlerine göre kelimelere ayırır. Kelime ayrıştırma işleminde beş farklı alan kullanır: komut, kaydedici, etiket, değer ve açıklama.
- Ayrıştırılan kelime, kelime alanının çeşidine göre renklendirilmektedir. Bu işlem kodları okumayı ve hataları görebilmeyi kolaylaştırmaktadır.
- Ayrıştırılan kelimeler kelime, tür ve değer olarak bir dizide biriktirilmektedir. Bu dizideki kelimeler genel satır değerlendirmesinden sonra hatasız ise parametre değerlendirmesinden geçirilmektedir. Hatalı durumlar kullanıcıya bildirilmekte ve satır derlemeye dahil edilmemektedir.
- Genel satır değerlendirmesinde assembly dili yazım kurallarına uygunluk kontrol edilmektedir.
- Parametre kontrolünde satırda bulunan komuta uygun sayıda ve nitelikte işlenen kullanılıp kullanılmadığı kontrol edilmektedir.
- Derleme yapılıyorsa hatalı satırlar hata numaralarıyla birlikte bir diziye kaydedilmektedir.
- Seçime göre metin kutusunda imlecin üzerinde bulunduğu satıra ait hata ya da satırda kullanılan komutun kullanımı ile ilgili bilgi, araç çubuğunun sağ tarafında yer alan bilgi kutusunda görüntülenmektedir.

### 6.2.2. Derleme hata mesajları

Editör penceresinde yazılan programın makine diline çevrilmesi için Assembler butonuna basıldığında, seçime göre programdaki hatalı tüm satırlar ile ilgili araç

çubuğundaki bilgi kutusunda hata mesajları verilmektedir.

Editör penceresinde imlecin üzerine hareket ettiği satır hata kontrolünden geçirilerek hata olması durumunda kullanıcıya bildirilmektedir. Programın baş kısmına list yada include satırlarını yazmaya gerek olmadan “trisa”, “w”, “rp1” ve diğer PIC16F84 için tanımlanmış sabitleri editör tanımaktadır (Şekil 6.3).



Şekil 6.3 Editörde komutların ayrıştırılması

Programcının yazdığı kodların makine diline çevrilmesi bir dizi kurallar çerçevesinde gerçekleşir. Bu kurallar çeviriciyi (assembleri) yazan kişilerce belirlenir. PIC16F84 mikrodenetleyici simülatörüne has assemblerde oluşabilecek hatalar satır ve parametre hataları olmak üzere ikiye ayrılmaktadır. Bu başlıklar altında kullanıcıya verilen hata mesajları aşağıda verilmektedir.

Genel satır hataları:

1. Bir komut satırı en fazla 4 kelime olmalıydı
2. İlk kelime komut yada etiket olmalıydı
3. Bir komut satırında yalnızca bir tane komut olur!
4. Bu komuta ait birinci parametre bit düzeyinde olamaz!
5. GOTO ve CALL harici komuttan sonra etiket olamaz!



6. İlk iki kelimedenden biri mutlaka komut olmalıydı.
7. Bu komuta ait ikinci parametre file kaydedici olamaz!
8. İlk kelime komut olduğundan satır 3 kelimedenden fazla olamaz!

Parametre Hataları: Komutlar kullandıkları parametre yapısına göre parametresiz, bir parametrelili ve iki parametrelili olarak üçe ayrılmaktadır. Kontrol edilen satırda parametre hatası araç çubuğundaki bilgi kutusunda programcının doğru kullanımını sağlayacak bilgilendirme olarak verilmektedir.

Parametre hatası mesajları:

1. Bu komutun hiçbir parametresi olmamalı
2. Bu komutun tek ve birinci parametresi 0-255 arası bir değerdir. Goto – Call komutları için 0-1023 arası bir değerdir.
3. Bu komutun tek ve birinci parametresi file register olmalı. File register adresi (Hex 00-4F) 0-79 / (Hex 80-CF) 128-207 arası olmalıdır.
4. Bu komutun birinci parametresi file register olmalı. File register adresi (Hex 00-4F) 0-79 / (Hex 80-CF) 128-207 arası olmalıdır. İkinci parametresi 0-7 arası bir değer olmalı.
5. Bu komutun birinci parametresi file register olmalı. File register adresi (Hex 00-4F) 0-79 / (Hex 80-CF) 128-207 arası olmalıdır. İkinci parametresi 0 yada 1 olmalı.

### 6.3. Hata Ayıklama ve Test Etme

Bu noktaya kadar yapılan işlem, editör kullanarak assemblerin kabul edeceği komutlar dizisini girmektir. Programcı editörde yazdığı programı önce derle butonuna basarak assembly diline uygunluğunu kontrol eder. Programı yapı bakımından oluşabilecek hatalardan arındırmak için hata ayıklama araçları tasarlanmıştır. Bu araçların yapabildikleri işlemler aşağıda sunulmaktadır:

- Programda kritik satırlarda durma noktası koyabilme,
- Seri çalışma dışında adımlayarak çalışma,

- Çalışmayı durdurma,
- Özel dosya kaydedicilerini içeren RAM bellek, EEPROM bellek ve yığın içeriklerini sıfırlayarak programı yeniden başlatma,
- İstenilen noktada özel dosya kaydedici, RAM bellek, EEPROM bellek ve yığın içeriklerinin dökümünü görebilme,
- Hata sezmesi durumunda kullanıcıya programa kalındığı yerden devam edebilme,
- Özel dosya kaydedicilerinin içeriklerini inceleme,
- RAM ve EEPROM bellek içeriklerini inceleme imkanları vermesi şeklinde sıralanabilir.

Derlenmiş programın çalışması anında, kaydedici ve bellek içeriklerinin gösterimi hızlı olmasından dolayı kullanıcı tarafından takibi zor olacaktır. Bu nedenle, simülatöre hata ayıklamada kullanılmak için işlevleri aşağıda verilmiş dört buton bir hız çubuğu eklenmiştir (Şekil 6.4).



Şekil 6.4 Program çalıştırma seçenekleri

- Hız çubuğu ile seri çalıştırma modunda iken bir komut satırı çalıştırıldıktan sonra ikinci komut satırını işlemeye almadan önce beklenecek süreyi ayarlamaktadır. Hız çubuğu 0-5000msn arasında 250msn'lik aralıklarla ayarlanabilmektedir. Hız çubuğunda yapılan ayar animasyon penceresine etki etmemektedir. Animasyon penceresindeki iç çalışma kendi sabit hızında otomatik olarak yada Q palsi butonu vasıtasıyla adım-adım yürütülmektedir.
- Yeşil trafik lambalı buton programı kaldığı yerden seri olarak çalıştırmaktadır. Durdurma butonuna basıldığından yada işletilen komut satırında durma noktası olduğundan program çalışması durmuşsa bu butona basılarak seri çalıştırılma başlatılabilir.
- Kırmızı trafik lambalı buton programın seri çalıştırılması esnasında istediğimiz bir noktada durdurmak için kullanılır. Durma noktalarını kullanıcı önceden koymasına karşılık burada durdurma seçimi o anlık gerçekleşir.

- Adımlı buton ilk basıldığında adımlayarak çalışmayı başlatır ve her basıldığında sıradaki komut satırını çalıştırır.
- Prg butonu özel dosya kaydedicilerinde bulunduğu RAM belleği, EEPROM belleğin ve yığının içeriklerini sıfırlar ve programın çalıştırıcı sırasını ilk başlangıca yönlendirir.
- MCLR reset butonu ise programın çalıştırıcı sırasını değiştirmeksizin özel dosya kaydedici içeriklerini PIC mikrodenetleyiciye ilk enerji verildiği duruma getirir.

Hata ayıklama aracı olarak yukarıda verilen çalışma butonları dışında içerikleri inceleyebilme için sunulan üç tip pencere mevcuttur. Bu pencereler:

- İşlem kütüğü penceresi,
  - Programın derlenmiş penceresi,
  - Bellek pencereleri (RAM, EEPROM, program belleği ve yığın).
- İşlem kütüğü penceresi, her komut saykılında, mikrodenetleyicide icra edilen komut satırını, özel ve genel amaçlı dosya kaydedicileri, yığın ve EEPROM'da yaptığı değişiklikleri incelemek amaçlı bir ekrandır. Özel dosya kaydedici içerikleri hem onaltılık düzende hem 8-bit olarak bit düzeninde gösterilmektedir (Şekil 6.5). Her bit kutucuğunun üstüne fare ile tıklandığında o bitin işlevi ile ilgili araç çubuğundaki bilgi kutusunda bilgi verilmektedir.

İşlem Kütüğü			
Adr Ad	Hex - Binary Değerler	Adr Ad	Hex - Binary Değerler
01 TMR0	02 0 0 0 0 0 0 0 1 0	0A PCLATH	02 0 0 0 0 0 0 0 1 0
02 PCL	07 0 0 0 0 0 0 1 1 1	0B INTCON	74 0 1 1 1 1 0 1 0 0
03 STATUS	19 0 0 0 1 1 0 0 1 1	81 OPTION	FF 1 1 1 1 1 1 1 1 1
04 FSR	00 0 0 0 0 0 0 0 0 0	85 TRISA	1F 0 0 0 1 1 1 1 1 1
05 PORTA	00 0 0 0 0 0 0 0 0 0	86 TRISB	FF 1 1 1 1 1 1 1 1 1
06 PORTB	E9 1 1 1 0 0 0 0 0 0	88 EECON1	00 0 0 0 0 0 0 0 0 0
08 EEDATA	00 0 0 0 0 0 0 0 0 0	89 EECON2	00 0 0 0 0 0 0 0 0 0
09 EEADR	00 0 0 0 0 0 0 0 0 0		

Şekil 6.5 İşlem kütüğü penceresi özel dosya kaydedicileri kısmı

İşlem kütüğü penceresi iki kısımdan oluşmaktadır. Pencerenin sol kısmında özel dosya kaydedicileri bulunmaktadır. Pencerenin sağ kısmının en üst satırında o anda

icra edilmekte olan komutun gerçekleştirdiği değişiklikleri gösterirken geriye yönelik olarak tüm işlenmiş satırların gerçekleştirdiği değişiklikleri hemen altında gösterilmektedir (Şekil 6.6).

Is No	PC	Komut	W	01	02	03	04	05	06	08	09	0A	0B	81	85	86	88	89	Bel	O/Y	Adr	Veri	
10	0206	MOVL A		FF	02	07	18	00	00	E0	00	00	02	74	FF	1F	FF	00	00	Y	YAZ	0	0204
2	0200	MOVL A		FC	02	01	18	00	00	00	00	00	02	00	FF	1F	FF	00	00				
3	0201	MOV A, F		FC	FC	02	18	00	00	00	00	00	02	00	FF	1F	FF	00	00				
4	0202	MOVL A		F0	FC	03	18	00	00	00	00	00	02	00	FF	1F	FF	00	00				
5	0203	MOV A, F		F0	FD	04	18	00	00	00	00	00	02	F0	FF	1F	FF	00	00				
6	0204	ADD A, F		F0	FE	05	18	00	00	F0	00	00	02	F0	FF	1F	FF	00	00				
7	0205	GOTO		F0	FF	04	18	00	00	F0	00	00	02	F0	FF	1F	FF	00	00				
8	0204	ADD A, F		F0	00	04	18	00	00	E0	00	00	00	74	FF	1F	FF	00	00	Y	YAZ	0	0204
9	0004	GOTO		F0	01	05	18	00	00	E0	00	00	02	74	FF	1F	FF	00	00				

Şekil 6.6 İşlem kütüğü penceresi dosya kaydedicileri kısmı

Pencerenin en sağında komutun genel amaçlı dosya kaydedicileri, yığın ya da EEPROM belleğinde yapmış olduğu değişiklikler gösterilmektedir. Burada “Bel” başlığı altında kullanılan Y (Yığın), E (EEPROM) ve R (RAM) anlamına gelmektedir.

İşlem kütüğünde, işlenen komutun içeriğini değiştirdiği özel kaydediciler renklendirilerek komutun etkilediklerinin fark edilmesi sağlanmaktadır.

Programın derlenmiş penceresinde, derleme işlemi sonucu hatasız bulunan satırlar, toplu halde gösterilmektedir. Adres kısmında belirlenen adrese tek tıkladığında adres kırmızı ve ikinci bir tıklamada adres yeşil hale gelmektedir. Kırmızı haldeki adres durma noktasıdır ve çalıştırma sırası o adrese geldiğinde çalışma otomatik olarak durmaktadır. Programın çalışması, “adım” ya da “çalıştır” butonuna basılarak devam ettirilmektedir.

Komut satırlarının bir altprograma daldığı veya G/Ç portlarından giriş ya da çıkış yaptığı önemli noktalarına durma noktaları konularak buralardaki yazılım-donanım ilişkisi ve kaydedici içerikleri kayıt altına alınabilir [14].

Bellek penceresi program, RAM, EEPROM ve yığın belleklerinin dördüne birden bakılabilecek bir pencere sunmaktadır. Gerçekleştirilen animasyonlarda tüm bellek dizisine bakılabilmektedir. İstenilen bir bellek dizisine hızla bakılabilmesi için,

adresin yazılması ve git butonuna basılması yeterli olmaktadır. PIC16F84 komutları 14 bit olduğundan program ve yığın bellekleri onaltılık sayı düzeninde 4 basamaktır. Aşağıdaki şekilde 0200 adresli program belleğinde derlenmiş bir programın “hex” kodları görülmektedir (Şekil 6.7).

Bellekler							
RAM		EPPROM		PROGRAM		YIĞIN	
Adr Değ		Adr Değ		Adr Değ		Değ	
00	FF	00	FF	0200	33FC	1	0204
00	FF	00	FF	0201	0081	2	0000
00	FF	00	FF	0202	33F0	3	0000
00	FF	00	FF	0203	008B	4	0000
00	FF	00	FF	0204	0786	5	0000
00	FF	00	FF	0205	2BFF	6	0000
00	FF	00	FF	0206	33FF	7	0000
00	FF	00	FF	0207	0086	8	0000
00	FF	00	FF	0208	0009		
00	FF	00	FF	0209	0000		

Şekil 6.7 Program belleğinde derlenmiş bir programın gösterilişi

EEPROM bellek penceresi 64 baytlık EEPROM’un tüm bellek içeriğini göstermektedir (Şekil 6.8). Programın icrası esnasında içeriği değişecek EEPROM bellek gözü bir animasyonla yanıp söndükten sonra içeriği değişmektedir.

EEPROM																
	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
0000	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0010	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Şekil 6.8 EEPROM bellek penceresi

Özel dosya kaydedicilerini de üzerinde barındıran 256 baytlık RAM bellekte kullanılmayan bölgeler gri olarak tüm bellek içeriğini göstermektedir (Şekil 6.9). Programın icrası esnasında içeriği değişecek RAM bellek gözü bir animasyonla yanıp söndükten sonra içeriği değişmektedir.

Hata ayıklama işlemleri, mikrodenetleyici simülatöründe grafiksel ortamda çok rahat bir şekilde yapılmaktadır.

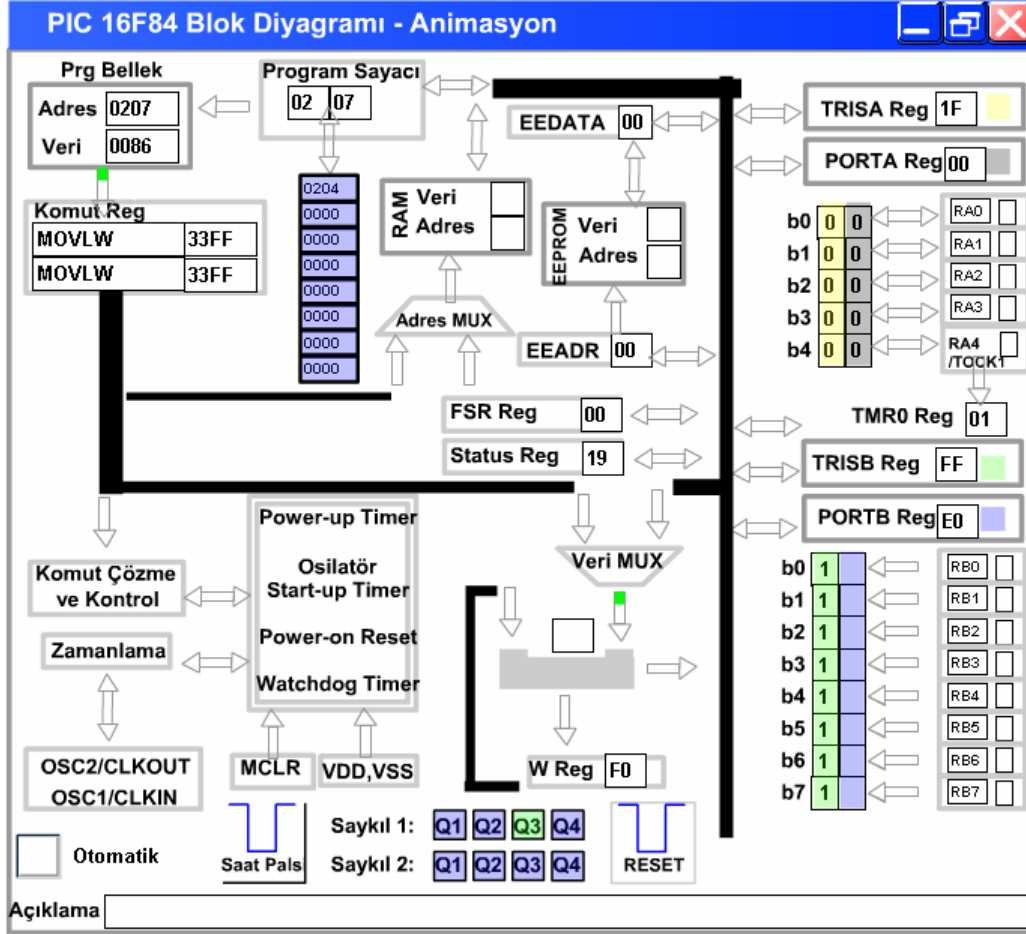
RAM																
	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
0000	00	02	05	15	00	00	E0	00	00	00	02	74	00	00	00	00
0010	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0080	00	FF	05	15	00	1F	FF	00	00	00	02	74	00	00	00	00
0090	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Şekil 6.9 RAM bellek penceresi

#### 6.4. Animatör

Bir mikrodenetleyici simülatörü en basit şekilde editör, derleyici ve hata ayıklama araçlarından meydana gelmektedir. Bu araçlar kullanıcının programını assembly diline uygun olarak kolayca yazabilmesi ve derlemesini sağlar. Aynı şekilde bu programlar kullanıcının, mikrodenetleyici mimarisinin parçası olan özel dosya kaydedicileri, RAM-EEPROM belleklerde meydana gelmesi istenilenlerin gözlemlenip etkilenebilmesini de sağlar. Ancak bütün bu işlemler sırasında veri yollarının durumu, zamanlama ve kontrol biriminin etkisi ve en önemlisi çalıştırılan program eğer bir G/Ç cihazı üzerinde denetim yapıyorsa o andaki etkilerinin görülmesi açısından simülatör tek başına yeterli olmayabilir. Bu düşünceden yola çıkılarak simülatöre Şekil 6.10'da görülen, mikrodenetleyici, EEPROM bellek, G/Ç birimleri ve iletişim yolları üzerindeki veri hareketlerinin gözlenebileceği bir animatör eklenmiştir.

Geliştirilen animatör, PIC16F84 mikrodenetleyicili bir sistemde mikrodenetleyicinin kullandığı tüm komutların mikrodenetleyici mimarisinin içerisinde çalışmasını grafiksel olarak göstermektedir. RAM bellek ve EEPROM bellek mimari içinde adres ve içerdiği veri olarak gösterilmektedir.



Şekil 6.10 PIC16F84 mikrodnetleyici animasyon penceresi

PIC16F84 Mikrodnetleyici komut setinde 35 komut bulunmaktadır. Bu komutlardan her birisi farklı bir işlemi gerçekleştirir. Bir komutun icrası/işlenmesi sırasında PIC16F84 mikrodnetleyicisinde gerçekleştirilen en küçük işlem zamanı saat palsinin 4 parçaya bölüdüğü Q (Q1, Q2, Q3 ve Q4) sistem saykılıdır.

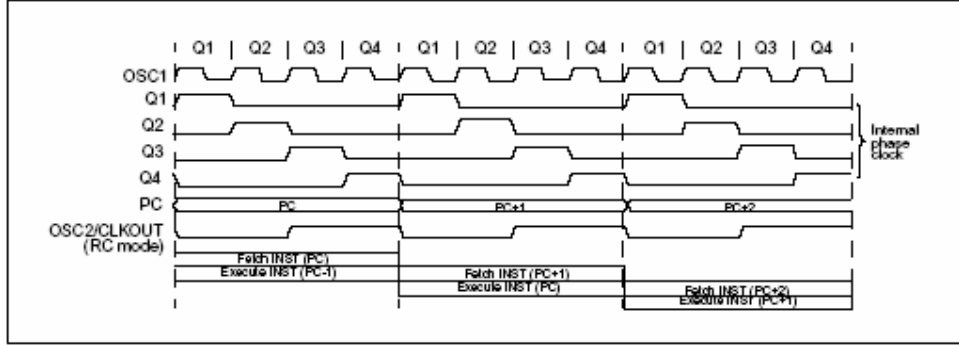
Komutun canlandırılması sırasında gerçekleşmekte olan Q palsin zemin rengi yeşil yapılmaktadır. Pencere üzerinde bulunan reset butonu ile işlenmekte olan komuta ait canlandırma tekrarlanabilmektedir.

#### 6.4.1. Komut saykılı ve boru akışı

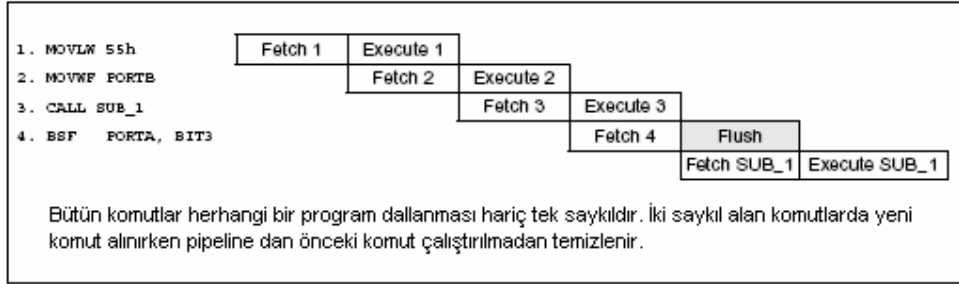
İşlemciye OSC1'den sağlanan saat palsleri içeride, Q1, Q2, Q3 ve Q4 olarak adlandırılan ve birbirleri ile örtüşmeyen kare dalgalara bölünür. Her bir Q1 palsinde

program sayacı (PC) artırılır, program belleğinden komut alınır ve Q4 palsinde IR1 komut kaydedicisine tutturulur. Komut Q1'den Q4 boyunca çözülür ve çalıştırılır (Şekil 6.11).

#### KOMUT SAYKILI



#### KOMUT BORU AKIŞI



Şekil 6.11 Mikrodenetleyici komut saykılı ve komut boru akışı

Bir komut saykılı 4 farklı Q saykılından (Q1, Q2, Q3 ve Q4) oluşur. Komutu alma ve çalışma ayrı birimlerde eş zamanlı yapılmaktadır. Q1 saykılında IR1 komut kaydedicisinde tutulan komut IR2 komut kaydedicine aktarılır ve oradan kod çözücüye aktarılarak çalıştırılır. Bu esnada yeni komutu almak için PC sayacı bir artırılır ve yeni komut Q4 saykılında IR1'e tutturulur.

Komut tutma adımları:

- Q1: PC artırılır.
- Q2: Program belleği seçilir.
- Q3: İşlenecek komut IR1'e atılır.
- Q4: IR1deki komut IR2'ye atılır.

Komut çözme ve çalışma: Çoğu komutta benzer sırada ve yapıda olmakla birlikte



bazı komutlarda farklı çalışmaktadır.

Q1 saykılında IR2'den işkodunun farklı parçaları aynı anda kod çözücü, adresmux ve datamux'a gider. Ancak adresmux yada datamux henüz kod çözücü-kontrol devresinden seçme sinyali almadığından işlem yapmayacaktır.

Adresmux'un seçilmesi durumunda işlenen işkoduna göre (kod çözücü-kontrol devresinin sinyaliyle) iki alternatif vardır. Doğrudan adresleme yapılmakta ise adresmux'a IR2den gelen 7 bitlik adres bilgisi ve durum kaydedicisinden gelen bank seçme (RP0) biti ile RAM adres yolu üzerinden RAM'e iletilerek RAM adreslenir. Dolaylı adreslemede ise FSR kaydedicisinin içeriği adresmux üzerinden RAM adres yoluna geçirilecek ve oradan RAM'e iletilerek adresleme yapılmış olacaktır. Kontrol uçlarından biri RAM'i oku/yaz yönünde tetiklerken bir diğeri (aynı zamanda) adresmux'u tetikleyerek adresmux girişlerinden birinin RAM seçme girişlerine iletilerek RAM'i adreslemesini sağlar. RAM'de adreslenen bölgede RAM'e kontrol uçlarından gelen oku/yaz sinyaline göre okuma ya da yazma işlemi gerçekleşir.

Datamux'un seçilmesi durumunda ise işlenen işkoduna göre (kod çözücü-kontrol devresinin sinyaliyle) datamux'a IR2'den gelen 8 bitlik bilgiyi ya da RAM den gelen 8 bitlik bilgiyi alır ve ALU birimine işlem yapmak üzere iletir.

Standart olarak aynı yapıda gerçekleştirilen iki farklı komut grubunun kod çözümü aşağıda sunulmaktadır.

LW (k): ANDLW – IORLW – XORLW – ADDLW – SUBLW

Q1: IR2'de tutulan komut kod çözücüye gönderilir.

Q2: İşkodu içerisinde gelen 8 bitlik sabit sayı datamux'a gönderilir.

Q3: Datamux seçilir ve kendisine gelen 8 bitlik sabit sayı bilgisini ALU birimine iletir.

Q4: ALU da işlem sonucu elde edilen sonuç W kaydedicisine aktarılır.

(f,d):COMF – DECF – INCF – RLF – RRF – SWAPF – DECFSZ – INCFSZ-

ANDWF – IORWF – XORWF – ADDWF – SUBWF - CLRF – BCF – BSF

Q1: IR2’de tutulan komut kod çözücüye gönderilir.

Q2: Adresmux seçilerek IR2’den gelen 7 bitlik adres bilgisi ile durum kaydedicisinin bank seçme biti RP0 bilgisini RAM’i adreslemede kullanır. RAM’de seçilmiş olan kaydedici içeriği veriyoluna bırakılır.

Q3: Datamux seçilir ve veri yolu üzerinden gelen kaydedici içeriğini ALU birimine iletir.

Q4: ALU’da ki işlem sonucu elde edilen sonuç komutta yer alan d bitine göre W ya da veri yolu üzerinden RAM de seçili özel kaydedici içeriği değiştirilir.

Q4 saykılında ALU’da yapılan işlem komuta göre değişecektir. Ancak blok diyagram üzerinden sadece ALU’da işlem yapıldığı gösterilmektedir. ALU’da işlemi yapılmakta olan komutlar şunlardır:

- Shift: RLF – RRF – SWAPF – COMF
- Logical: ANDLW-ANDWF-IORLW-IORWF-XORLW-XORWF
- Addition-Subtraction: ADDLW-ADDWF-SUBLW-SUBWF – DECF – INCF
- CLRF - CLRW- BSF – BCF

İşlenmesi esnasında ikinci saykıl gerektiren komutlar: CALL -GOTO -RETFIE-RETLW-RETURN

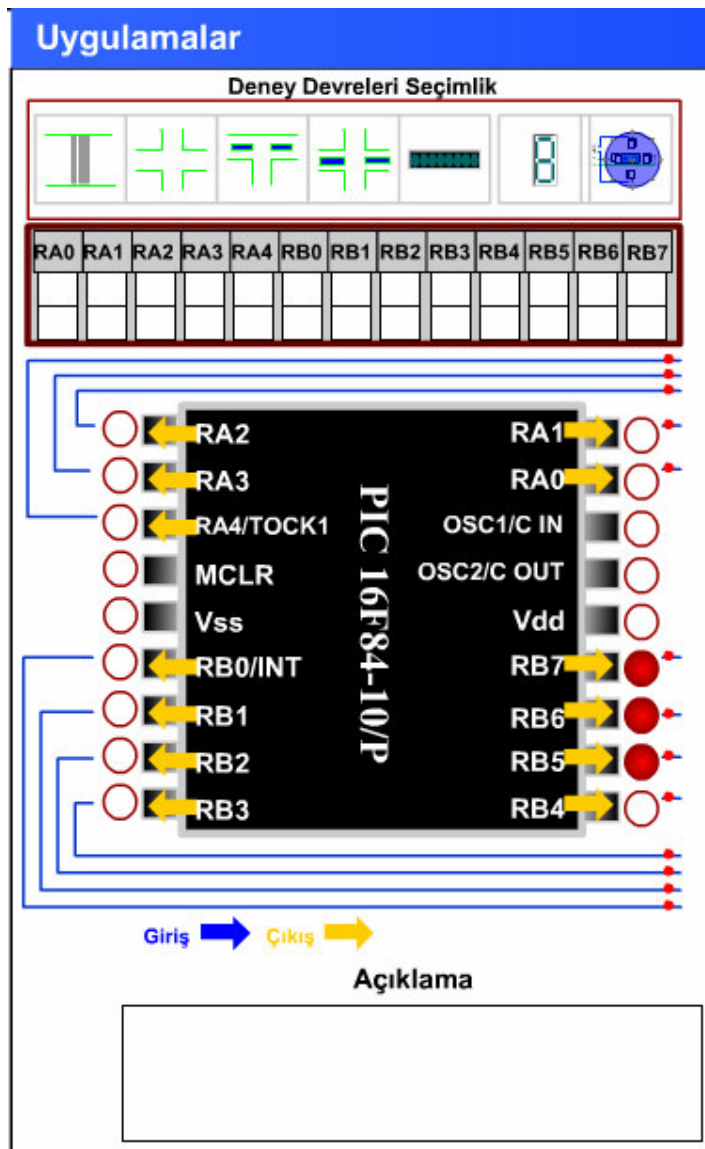
İşlenmesi esnasında şarta bağlı ikinci saykıl gerektiren komutlar: BTFSC- BTFSS-DECFSZ-INCFSZ

İkinci saykıl boyunca dört palste de kod çözme kısmı için hiçbir işlem yapılmayacaktır.

## 6.5. Deney Devreleri ve Kullanımı

Kullanıcı assembly dilinde bir program yazar ve derleme işlemine tabi tutar. Kullanıcı, derlenmiş programın çalıştırılması sonucu özel dosya kaydedicileri, RAM bellek, EEPROM bellek, ALU ve PIC16F84 portları üzerindeki etkilerini görür.

Animatörde ise derlenmiş programın çalıştırılması sonucu veri yollarının durumu, zamanlama ve kontrol biriminin etkisi ve G/Ç cihazı üzerindeki denetim etkilerini görür. Ancak tüm bunlar günlük hayatta mikroişlemcinin hangi çevre cihazlarıyla ne gibi denetimler yapmakta olduğunu görebilme ve deneme yapabilme imkanı vermez. Bu düşünceden yola çıkarak simülatöre Şekil 6.12’de sol parçası görülen, mikrodenetleyici, mikrodenetleyici portlarının bağlı olduğu çevre birimlerini gösteren tablo ve çevre birimlerinin örnek uygulamalarından oluşan deneyler penceresi eklenmiştir.



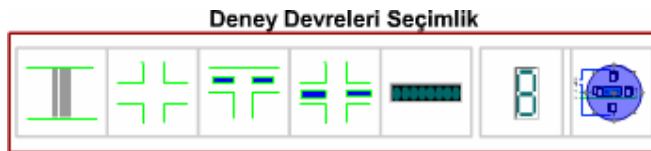
Şekil 6.12 PIC16F84 mikrodenetleyici deney uygulamaları penceresi

Deneyler penceresinde PIC16F84 RA0-RA4 ile RB0-RB7 uçlarına bağlantı kurulabilmektedir. Bağlantı kurulabilen bu uçlara komut satırının işlenmesi sonucu ya da bağlı anahtarlar dolayısıyla gelen değer lojik “1” ise entegre ayağının yanındaki daire kırmızı renk yanmakta aksi durumda içi boş görünmektedir. Port ucu giriş yönünde kurulu ise ucun üzerinde bulunan ok mavi ve yönü port girişine doğru, çıkış yönünde kurulu ise ok portakal renkli ve yönü port ucuna ters duruma getirtilerek kullanıcının port uçlarının giriş/çıkış yönünde kurulu olduğunu anlaması sağlanmaktadır.

Kullanıcı istediği PIC16F84 çıkış portlarına fare ile tıkladıktan sonra deney modülündeki trafik lambası ise seçilecek lambasına, 8-bitlik LED ise seçilecek LED’e, yedi parçalı gösterge ise seçilecek LED’e ve adım motoru ise ayak bağlantılarına tıklamak suretiyle bağlantı kurulur. Bağlantılar doğrudan PIC16F84 portlarından düzenlenmiş hat üzerinden bağlantı kurulacak elemana doğru diğerlerinden farklı renkte çizgi ile gösterildiği gibi deneyler penceresinin üst kısmındaki listede de gösterilmektedir.

Deneyler penceresinin sağ üst kısmında uygulama modülü seçme işleminin yapıldığı seçimlik kısmı gösterilmektedir (Şekil 6.13). Simülatör programlarının en büyük artısı sayılabilecek özelliklerinden birisi; yeni modüllerin eklenmesinin kolay olmasıdır. Seçilebilecek 7 uygulama parçası bulunmaktadır.

Uygulama parçaları; dört adet, trafik ışık kontrolü, bir adet 8 bitlik LED grubu, 1 adet yedi parçalı gösterge ve 1 adet adım motorundan oluşmaktadır (Şekil 6.13). Adım motoru dışındaki tüm elemanlar gerçekte LED mantıklı elemanlardır.



Şekil 6.13 Uygulama modülü seçim penceresi

### 6.5.1. Trafik ışık kontrolü uygulamaları

Kullanıcının günlük hayatta örnekler üzerinde çalışmasını sağlamak amacıyla 8085 mikroişlemcili sistemde deney uygulamaları için tasarlanan trafik ışık kontrolü uygulama modülleri PIC16F84 mikrodenetleyicisi deney uygulamalarında aynı şekilde ve yapıda kullanılmaktadır.

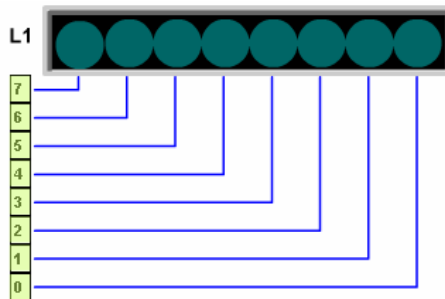
Trafik ışık kontrolü uygulamaları, kullanıcıya trafik lambalarındaki kırmızı, sarı ve yeşil ışıkların yanış sürelerinin programlanmasını öğretmek içindir. Trafik lambaları, gecikme amaçlı programların geliştirilmesinde en etkin araçtır.

Trafik lambalarından çıkarılan uçlar PIC16F84 portlarından düzenlenmiş hatta yanaşık düzende olması sağlanmıştır. Kullanıcı bağlantı kurmak istediği PIC16F84 ayağına fare ile tıkladıktan sonra trafik lambalarından istediğinin herhangi renkteki lambasına ya da lambadan çıkarılarak bir hat üzerine getirilen uç noktasına tıklayarak bağlantıyı kurmaktadır.

### 6.5.2. 8-bitlik ledler

Gerek elektromekanik gerekse simülâtör temelli öğretim araçlarında en kullanışlı kontrol elemanı LED'lerdir. Görsel, ucuz ve küçük ebatlı olmalarından dolayı elektronik sistemlerde durum işaretçisi olarak sık kullanılmaktadır [14].

Kullanıcı bağlantı kurmak istediği PIC16F84 ayağına fare ile tıkladıktan sonra istediği bir led'e yada led'den bir bağlantı ile çıkartılarak düz bir hat üzerine konulmuş uca tıklayarak bağlantı kurabilmektedir (Şekil 6.14).

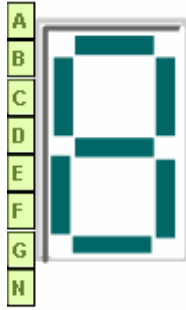


Şekil 6.14 8-Bitlik LED paneli

Uygulamalarda, dizi şeklindeki LED’ler üzerinde birçok sanal işlem gerçekleştirilebilir. Bunlar, LED’lerin sırasıyla sağa ve sola doğru yakılması, ileri veya geri saydırılması olabilir.

### 6.5.3. Yedi parçalı gösterge uygulaması

Elektronik sistemlerde veri akışının basit yöntemle gözlenmesini sağlayan yedi parçalı gösterge simülatör içinde geliştirilmiştir. Her bir parçası yedi adet LED’ten meydana geldiğinden üzerinde basit bilgiler gösterilebilmektedir (Şekil 6.15). Bu kolaylıktan faydalanarak, gösterge üzerine çeşitli harf veya sayılar yazdırılarak kaydırılabilmektedir [14].



Şekil 6.15 Yedi parçalı gösterge uygulaması

### 6.5.4. Adım motoru

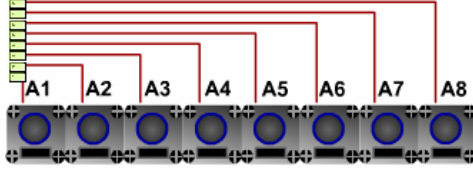
En basit şekliyle disket sürücülerin içerisinde bulunan adım motorları yaygın kullanıma sahip elektromekanik bir elemandır.

Bölüm 5’de 8085 mikroişlemcili sistemde deney uygulamaları için tasarlanan adım motoru uygulama modülü PIC16F84 mikrodenetleyicisi deney uygulamalarında aynı şekilde ve yapıda kullanılmaktadır.

### 6.5.5. Anahtarların giriş elemanları olarak kullanılması

Simülatörde çevre giriş elemanı olarak bir adet anahtar grubu kullanılmıştır (Şekil 6.16). Anahtarlar üzerinde biri daire, diğeri dikdörtgen şeklinde iki buton yer almaktadır. Daire şeklinde olanına basılı tuttuğunuz müddetçe çalışırken, dikdörtgen

şeklinde olan butona tıklanarak anahtar basılı pozisyonda tutulmakta ya da tekrardan serbest bırakılmaktadır.



Şekil 6.16 Anahtar grubu

## **BÖLÜM 7. SONUÇ ve ÖNERİLER**

Mesleki ve Teknik Okullarda yıllardan beri ders olarak okutulan Mikroişlemciler / Mikrodenetleyiciler derslerinde öğrencilerin oldukça zorlandıkları görülmektedir. Bu zorluklar mikroişlemcili sistemlerin öğretiminin genelde sınıflarda kuramsal olarak veya laboratuarlarda yetersiz elektromekanik eğitim setleri üzerinde kısa süreli verilmesinden kaynaklanmaktadır.

Gerçekleştirilen grafik tabanlı fonksiyonel mikroişlemci / mikrodenetleyici simülatörü, görsellik ve esneklikten yoksun geleneksel elektromekanik eğitim setlerine göre birçok avantaja sahiptir. Tüm işlevsel birimler ve kolaylıklara sadece bir fare tıklamasıyla erişilebilir.

Mikroişlemci / mikrodenetleyici simülatörü, üzerinde yerleşik arabirim olan editörde kullanıcının assembly dilinde program yazmasını sağlamaktadır. Editörün iki önemli hususu vardır: ilki kullanıcının kullandığı komutların kullanımı hakkında yardım verebilmesi, ikincisi hatalı kod satırları için hata uyarıları vermesidir. Editörde yazılarak derlenmiş program için hata ayıklama pencereleri sayesinde programın işleyişindeki problemlerin daha kolay giderilmesine imkan sağlanmıştır.

Simülatöre bellek, G/Ç birimleri ve iletişim yolları üzerindeki veri hareketlerinin gözlenebileceği bir animatör eklenmiştir. Bu animatörde komutların icrası esnasında mikroişlemci / mikrodenetleyici iç mimarisindeki çalışması anime ettirilerek gösterilmiştir.

Simülatöre, günlük hayatta, mikroişlemci / mikrodenetleyicinin çevre cihazlarıyla gerçekleştirdiği denetimleri görebilme ve deneme yapabilme imkanı verilen deneyler penceresi eklenmiştir.



Mikroişlemci / mikrodenetleyici mimarileri ile komutlar arasındaki ilişkilerin anlaşılması için, farklı kaydedicilerde komutların etkisinden doğan sonuçların gözlenmesi ve bellek yönetiminin anlaşılması sağlanmıştır.

Geleneksel eğitim setinde bir komutun çok hızlı bir şekilde işlenerek gerçekleştirdiği olayların gözlenememesi, simülâtörün tasarımında yatan programlama tekniği sayesinde ortadan kaldırılarak, kullanıcıya hız seçme seçenekleri sunulmuştur.

Simülâtör modüler yapıya sahiptir ve mikroişlemci / mikrodenetleyici için yeni çevre birimi modülleri eklenebilir. Geleneksel elektromekanik setlerde bunun yapılması hem güç hem pahalıdır.

Simülâtör “Macromedia Flash” programıyla hazırlandığından tüm bilgisayarlarda kurulum yapılmaksızın ve işletim sistemi ayırmaksızın çalıştırılabilmektedir. Flash programında hazırlanmış olmasının getirdiği iki özellik ile simülâtör, tüm bilgisayarlarda işletim sistemi ayırmaksızın çalıştırılabilmektedir. Simülâtör KBayt seviyesinde küçük bir dosya boyutuna sahip olduğundan dolayı web sayfası üzerinden sunulabilmekte veya öğrenci kendi bilgisayarına indirerek çalışabilmektedir. Böylece zamandan ve mekandan bağımsız çalışabilme imkanı doğmaktadır. Bunun sonucunda öğrenme süresi ve doğal olarak öğrenme artacaktır. Bütün bunların yanında geleneksel elektromekanik set alamayacak yada bunu kurabilecek mekanı bulunmayan kurumlar için hazırlanmış olan simülâtör iyi bir fırsattır.

Bundan sonraki çalışmada simülâtör, deney uygulamaları kısmına A/D ve D/A çevirici, motor hız sayıcı, alarm devreleri gibi sanal uygulama araçlarının eklenmesi ve animasyon kısmına ALU içinde gerçekleşen işlemlerin canlandırılmasının eklenmesi ile daha iyi bir öğretim aracı haline getirilecektir.

## KAYNAKLAR

- [1] Nesrin Özdener, Hamdi Sayın, “Macromedia Flash Eğitimi Amacı İle Geliştirilen Bir Eğitsel Yazılımın Bütünsel Ve Kullanılan Yöntemler Açısından Değerlendirilmesi”, Tojet, April 2004 (ISSN: 1303-6521 volume 3, issue 2, article 24)
- [2] İlhami Çolak, Erdal Irmak, İbrahim Sefa, Şevki Demirbaş, Ramazan Bayındır, “Açık ve Kapalı Döngü Denetim Sistemlerinin Web Tabanlı Benzetimi”, 6th International Educational Technology Conference, 19-21 Nisan 2006 Magusa, KKTC (Bildiriler Vol I sy 446)
- [3] Hüseyin Ekiz, Yavuz Bayam, Hüseyin Ünal, “Mantık Devreleri Dersine Yönelik İnternet Destekli Uzaktan Eğitim Uygulaması”, Tojet, October 2003 (ISSN: 1303-6521 volume 2, issue 4, article 14)
- [4] Hüseyin Ekiz, Aytaç Kaya, Hüseyin Ünal, Zafer Demir, “Mikroişlemci– Mikrodenetleyici Eğitime Yönelik İnternet Destekli Uzaktan Öğretim Uygulaması”, 6th International Educational Technology Conference, 19-21 Nisan 2006 Magusa, KKTC (Bildiriler Vol II sy 606)
- [5] Çolak İ., Irmak E., Demirbaş Ş., Bayındır R., “Teknik Eğitimde İnternet Teknolojisinin Kullanımı”, 1. Uluslar arası Mesleki ve Teknik Eğitim Teknolojileri Kongresi, 5-7 Eylül 2005, İstanbul
- [6] E.Akın, M. Karaköse, “Elektrik ve Bilgisayar Mühendisliği Eğitiminde Sanal Laboratuvarların Kullanımı”, “Elektrik, Elektronik, Bilgisayar Mühendislikleri Eğitimi, I. Ulusal Sempozyumu ve Sergisi”, Mayıs 2003, ODTÜ, Ankara, <http://egitim.emu.org.tr/bildiri.html>
- [7] Mevlüt Arslan, Ali Erişen, “Uzaktan Eğitimde Kullanılan Web Tabanlı laboratuvarlarda İnternet Kaynaklı Temel Sorunlar ve Çözüm Önerileri”, 6th International Educational Technology Conference, 19-21 Nisan 2006 Magusa, KKTC (Bildiriler Vol I sy 136)
- [8] Stallings W., Computer Organization and Architecture Desingning for Performance, 6 th Ed., Prentice Hall, 2003
- [9] Atakan Körez, A. Yılmaz Çamur, “Basit Bir Mikroişlemci Yapısının Web Tabanlı Çoklu Ortam İle Öğretimi”, IV. Uluslar arası Eğitim Teknolojileri Sempozyumu, 24-26 Kasım 2004 Sakarya, Türkiye (Bildiriler Vol I sy 104)

- [10] Halsal, F., Mikroelektronik sistemler (II), M.E.B. Çeviri Kitabı, Ankara, 1994
- [11] Milli Eğitim Bakanlığı, Projeler Dairesi, Endüstriyel Okullar Projesi Raporu, 1993
- [12] Nartkaya, M., Intel 8085 Mikroişlemci Simülasyonu, Yüksek Lisans Tezi , Afyon Kocatepe Üniversitesi, Fen Bilimleri Enstitüsü, 1996, Afyon
- [13] Yüksek Öğrenim Kurulu, Endüstriyel Eğitim Projesi Başkanlığı Raporu, 1997
- [14] Topaloğlu, N., PC Tabanlı Fonksiyonel Mikroişlemci Simülatör Tasarımı ve Gerçekleştirilmesi, Doktora Tezi, Gazi Üniversitesi, Fen Bilimleri Enstitüsü, 2002, Ankara
- [15] John D. Carpinelli, Fabio Jaramillo, “Simulation Tools For Digital Design and Computer Organization and Architecture”, 31st ASEE/IEEE Frontiers in Education Conference, 10-13 October 2001 Reno, NV
- [16] Alfredo del Rio, Juan Jose Rodriguez, Andres A. Nogueiras, “Learning Microcontrollers with a CAI-Oriented Multi-Micro Simulation Environment”, IEEE Transactions on Education, May 2001 Vigo, İspanya, (Vol 44 No 2)
- [17] Kurat, N., “İnternet Tabanlı PIC16F84 Eğitimi”, Yüksek Lisans Tezi, Gazi Üniversitesi, Fen Bilimleri Enstitüsü, 2002, Ankara
- [18] Taşkın, T., “MC6811 Microcontroller Simulation Toolkit”, Yüksek Lisans Tezi, METU, Fen Bilimleri Enstitüsü, 2005, Ankara
- [19] Altınbaşak, O., Mikrodenetleyiciler ve PIC Programlama, Altaş Basım Yayım Dağıtım, İstanbul, 2000.
- [20] Özkan, T., Arslan, S., 8085 Mikroişlemcili Mikroişlemci Deney Seti, Sakarya, 2005
- [21] Adalı, E., Mikroişlemciler Mikrobilgisayarlar, Birsen Yayınevi, İstanbul, 1998.
- [22] Ekiz, H., Mikroişlemciler / Mikrodenetleyiciler Ders Notu Dökümanları, Sakarya, 2005
- [23] Embedded Control Handbook, Microchip Technology Inc. [www.microchip.com](http://www.microchip.com) 2001
- [24] Akpınar, Y., Bilgisayar Destekli Eğitimde Uygulamalar, Anı Yayıncılık, Ankara, 2005
- [25] Nesrin Özden, “Deneysel Öğretim Yöntemlerinde Benzetişim (Simulation) Kullanımı ”, IV. Uluslar arası Eğitim Teknolojileri Sempozyumu, 24-26 Kasım 2004 Sakarya, Türkiye (Bildiriler Vol I sy 239)

- [26] Özkul, A., “E-Öğrenme ve Mühendislik Eğitimi”, Elektrik, Elektronik, Bilgisayar Müh. Eğitimi I. Ulusal Sempozyumu, Mayıs 2003, ODTÜ, Ankara, <http://egitim.emo.org.tr/bildiri.html>
- [27] Adnan KAKİLLİ, Caner AKÜNER, İsmail TEMİZ, “Güç Sistemlerinin Korunmasında Bilgisayar Destekli Simülasyon Uygulaması”, IV. Uluslar arası Eğitim Teknolojileri Sempozyumu, 24-26 Kasım 2004 Sakarya, Türkiye (Bildiriler Vol I sy 412)
- [28] Arslan, M., “İnternet Tabanlı Endüstriyel Sıcaklık Ölçme ve Kontrol Sistemi”, 2. Uluslar arası Mühendislik Ölçmeleri Sempozyumu, 23-25 Kasım 2005, İstanbul, (sy 124-132).
- [29] [www.flashdersleri.com/flash6.html](http://www.flashdersleri.com/flash6.html), 2005
- [30] Küçük, S., PIC16C65 Serisi Mikrokontrolörün PC’de Simülasyon İle Eğitimi, Yüksek Lisans Tezi, Marmara Üniversitesi, Fen Bilimleri Enstitüsü, 1998, İstanbul
- [31] Özkan, M.S., 68000 Mikroişlemcisinin PC’de Simülasyon ile Eğitimi (68000 Simülatörü), Yüksek Lisans Tezi, Marmara Üniversitesi, Fen Bilimleri Enstitüsü, 1995, İstanbul
- [32] Topaloğlu, N., X86 Tabanlı Mikroişlemci Mimarisi ve Assembly Dili, Seçkin Yayınevi, Ankara, 2001
- [33] Ekiz, H., “ECCB 6800 Mikroişlemci Kontrollü Deney Modülleri Tasarımı, Yapımı ve Mikroişlemci Setleri İçin Arıza Giderme Yöntemlerinin İncelenmesi”, Yüksek Lisans Tezi, Gazi Üniversitesi, Fen Bilimleri Enstitüsü, 1992, Ankara

## EKLER

### Ek-A 8085 MİKROİŞLEMCİSİ KOMUTLARI

Adresleme Türleri:

- R (Kaydedici)
- RI (Kaydedici Dolaylı)
- I (Dolaylı)
- IRI (İvedi Kaydedici Dolaylı)
- D (Doğrudan)

Komut özet tablosunda kullanılan semboller ve anlamları aşağıdaki gibidir:

- Kh: Hedef kaydedici
- Kk: Kaynak kaydedici
- D: Destination (Hedef kaydedici yada kaydedici çifti)
- S: Source (Kaynak kaydedici yada kaydedici çifti)
- M/<M>: Bellek
- V: 1 bayt (sekizli) veri
- VV: 2 bayt (sekizli) veri
- AA: 2 bayt (sekizli) adres
- (( )): kaydedici çiftine bağlı olan dolaylı adreslemeleri belirtir. Örneğin ((HL)) adresi HL kaydedici çiftinde bulunan bellek gözünü işaret eder.
- ←:sağdakinin kaydedici, veri, sıradaki bellek gözü yada kaydedici çiftinin gösterdiği bellek gözü içeriği soldaki kaydedici, sıradaki bellek gözü yada kaydedici çiftinin gösterdiği bellek gözüne aktarılımsın.

Komut	ADR	C	S	Bayrak	Opcode Çözümlenmeleri								Açıklama
MOV Kh,Kk	R	1	4	-	0	1	D	D	D	S	S	S	Kh←Kk
MOV Kh,M	RI	2	7	-	0	1	D	D	D	1	1	0	Kh←((HL))
MOV M,Kk	RI	2	7	-	0	1	1	1	0	S	S	S	((HL))←Kk
MVI Kh,V	I	2	7	-	0	0	D	D	D	1	1	0	Kh←V
MVIM,V	IRI	3	10	-	0	0	1	1	0	1	1	0	((HL))←V
LXI B,VV	I	3	10	-	0	0	0	0	0	0	0	1	BC←VV
LXI D,VV	I	3	10	-	0	0	0	1	0	0	0	1	DE←VV
LXI H,VV	I	3	10	-	0	0	1	0	0	0	0	1	HL←VV
LXI SP,VV	I	3	10	-	0	0	1	1	0	0	0	1	YG←VV
LDA AA	D	4	13	-	0	0	1	1	1	0	1	0	A←<M>
LDAX B	R	2	7	-	0	0	0	1	1	0	1	0	A←((BC))
LDAX D	R	2	7	-	0	0	0	1	1	0	1	0	A←((DE))
LHLD	D	5	16	-	0	0	1	0	1	0	1	0	Kh←<M,M+1>
SPHL	R	1	6	-	1	1	1	1	1	0	0	1	YG←HL
STA	D	4	13	-	0	0	1	1	0	0	1	0	M←A
STAX B	RI	2	7	-	0	0	0	0	0	0	1	0	((BC))←A
STAX D	RI	2	7	-	0	0	0	1	0	0	1	0	((DE))←A
SHLD	D	5	16	-	0	0	1	0	0	0	1	0	M←HL
IN	D	3	10	-	1	1	0	1	1	0	1	1	Port içeriğini A'ya aktar
OUT	D	3	10	-	1	1	0	1	0	0	1	1	A'nın içeriğini porta aktar
XCHG	R	1	4	-	1	1	1	0	1	0	1	1	H<->D L<->E
ADD Kk	R	1	4	SZAPC	1	0	0	0	0	S	S	S	A←A + Kk
ADC Kk	R	1	4	SZAPC	1	0	0	0	1	S	S	S	A← A + Kk + E
ADD M	RI	2	7	SZAPC	1	0	0	0	0	1	1	0	A←A + ((HL))
ADC M	RI	2	7	SZAPC	1	0	0	0	1	1	1	0	A← A + ((HL)) + E
ADI V	I	2	7	SZAPC	1	1	0	0	0	1	1	0	A←A + V
ACI V	I	2	7	SZAPC	1	1	0	0	1	1	1	0	A←A + V + E
DAD B	R	3	10	C	0	0	0	0	1	0	0	1	HL←HL + BC
DAD D	R	3	10	C	0	0	0	1	1	0	0	1	HL←HL + DE
DAD H	R	3	10	C	0	0	1	0	1	0	0	1	HL←HL + HL
DAD SP	R	3	10	C	0	0	1	1	1	0	0	1	HL←HL + YG
SUB Kk	R	1	4	SZAPC	1	0	0	1	0	S	S	S	A←A - Kk
SBB Kk	R	1	4	SZAPC	1	0	0	1	1	S	S	S	A←A - Kk - E
SUB M	RI	2	7	SZAPC	1	0	0	1	0	1	1	0	A←A - ((HL))
SBB M	RI	2	7	SZAPC	1	0	0	1	1	1	1	0	A← A - ((HL)) - E
SUI V	I	2	7	SZAPC	1	1	0	1	0	1	1	0	A←A - V
SBI V	I	2	7	SZAPC	1	1	0	1	1	1	1	0	A← A - V - E
INR Kh	R	1	4	SZAP	0	0	D	D	D	1	0	0	Kh← Kh + 1
INR M	RI	3	10	SZAP	0	0	1	1	0	1	0	0	((HL))← ((HL)) + 1

Komut	ADR	C	S	Bayrak	Opcode Çözümlenmeleri								Açıklama
INX B	R	1	6	-	0	0	0	0	0	0	1	1	BC← BC + 1
INX D	R	1	6	-	0	0	0	1	0	0	1	1	DE← DE + 1
INX H	R	1	6	-	0	0	1	0	0	0	1	1	HL← HL + 1
INX SP	R	1	6	-	0	0	1	1	0	0	1	1	SP← SP + 1
DCR Kh	R	1	4	SZAP	0	0	D	D	D	1	0	1	Kh← Kh + 1
DCR M	RI	3	10	SZAP	0	0	1	1	0	1	0	1	((HL))← ((HL)) - 1
DCX B	R	1	6	-	0	0	0	0	1	0	1	1	BC← BC - 1
DCX D	R	1	6	-	0	0	0	1	1	0	1	1	DE← DE - 1
DCX H	R	1	6	-	0	0	1	0	1	0	1	1	HL← HL - 1
DCX SP	R	1	6	-	0	0	1	1	1	0	1	1	SP← SP - 1
RLC	-	1	4	C	0	0	0	0	0	1	1	1	Sola döndür (C biti kat)
RRC	-	1	4	C	0	0	0	0	1	1	1	1	Sağa döndür (C biti kat)
RAL	-	1	4	C	0	0	0	1	0	1	1	1	Sola döndür
RAR	-	1	4	C	0	0	0	1	1	1	1	1	Sağa döndür
DAA	R	1	4	SZAPC	0	0	1	0	0	1	1	1	BCD çevir
CMA	-	1	4	-	0	0	1	0	1	1	1	1	A tümleyenini al
CMC	-	1	4	C	0	0	1	1	1	1	1	1	Elde bitini 0 yap
STC	-	1	4	C	0	0	1	1	0	1	1	1	Elde bitini 1 yap
ANA Kk	R	1	4	SZAPC	1	0	1	0	0	S	S	S	A← A . Kk
ANA M	RI	2	7	SZAPC	1	0	1	0	0	1	1	0	A← A . ((HL))
ANI V	I	2	7	SZAPC	1	1	1	0	0	1	1	0	A← A . V
XRA Kk	R	1	4	SZAPC	1	0	1	0	1	S	S	S	A← A (+) Kk
XRA M	RI	2	7	SZAPC	1	0	1	0	1	1	1	0	A← A (+) ((HL))
XRI V	I	2	7	SZAPC	1	1	1	0	1	1	1	0	A← A (+) V
ORA Kk	R	1	4	SZAPC	1	0	1	1	0	S	S	S	A← A + Kk
ORA M	RI	2	7	SZAPC	1	0	1	1	0	1	1	0	A← A + ((HL))
ORI V	I	2	7	SZAPC	1	1	1	1	0	1	1	0	A← A + V
CMP Kk	R	1	4	SZAPC	1	0	1	1	1	S	S	S	A ile Kk'yi karşılaştır
CMP M	RI	2	7	SZAPC	1	0	1	1	1	1	1	0	A ile M'yi karşılaştır
CPI V	RI	2	7	SZAPC	1	1	1	1	1	1	1	0	A ile V'yi karşılaştır
JMP AA	I	3	10	-	1	1	0	0	0	0	1	1	Şartsız bağlan
JC AA	I	2/3	7/10	-	1	1	0	1	1	0	1	0	Elde varsa bağlan
JNC AA	I	2/3	7/10	-	1	1	0	1	0	0	1	0	Elde yoksa bağlan
JZ AA	I	2/3	7/10	-	1	1	0	0	1	0	1	0	Sıfır ise bağlan
JNZ AA	I	2/3	7/10	-	1	1	0	0	0	0	1	0	Sıfır değilse bağlan
JP AA	I	2/3	7/10	-	1	1	1	1	0	0	1	0	Artı ise bağlan
JM AA	I	2/3	7/10	-	1	1	1	1	1	0	1	0	Eksi ise bağlan

Komut	ADR	C	S	Bayrak	Opcode Çözümlenmeleri								Açıklama
JPE AA	I	2/3	7/10	-	1	1	1	0	1	0	1	0	Eşlik biti çift ise bağlan
JPO AA	I	2/3	7/10	-	1	1	1	0	0	0	1	0	Eşlik biti tek ise bağlan
CALL AA	IRI	5	18	-	1	1	0	0	1	1	0	1	Şartsız dallan
CC AA	IRI	2/5	9/18	-	1	1	0	1	1	1	0	0	Elde varsa dallan
CNC AA	IRI	2/5	9/18	-	1	1	0	1	0	1	0	0	Elde yoksa dallan
CZ AA	IRI	2/5	9/18	-	1	1	0	0	1	1	0	0	Sıfır ise dallan
CNZ AA	IRI	2/5	9/18	-	1	1	0	0	0	1	0	0	Sıfır değilse dallan
CM AA	IRI	2/5	9/18	-	1	1	1	1	1	1	0	0	Artı ise dallan
CP AA	IRI	2/5	9/18	-	1	1	1	1	0	1	0	0	Eksi ise dallan
CPE AA	IRI	2/5	9/18	-	1	1	1	0	1	1	0	0	Eşlik biti çift ise dallan
CPO AA	IRI	2/5	9/18	-	1	1	1	0	0	1	0	0	Eşlik biti tek ise dallan
RET	RI	3	10	-	1	1	0	0	1	0	0	1	Ana programa geri dön
RC	RI	1/3	6/12	-	1	1	0	1	1	0	0	0	Elde varsa geri dön
RNC	RI	1/3	6/12	-	1	1	0	1	0	0	0	0	Elde yoksa geri dön
RZ	RI	1/3	6/12	-	1	1	0	0	1	0	0	0	Sıfır ise geri dön
RNZ	RI	1/3	6/12	-	1	1	0	0	0	0	0	0	Sıfır değilse geri dön
RM	R	1/3	6/12	-	1	1	1	1	1	0	0	0	Artı ise geri dön
RP	RI	1/3	6/12	-	1	1	1	1	0	0	0	0	Eksi ise geri dön
RPE	RI	1/3	6/12	-	1	1	1	0	1	0	0	0	Eşlik biti çiftse geri dön
RPO	RI	1/3	6/12	-	1	1	1	0	0	0	0	0	Eşlik biti tekse geri dön
PCHL	R	1	6	-	1	1	1	0	1	0	0	1	PS←HL
PUSH B	RI	3	12	-	1	1	0	0	0	1	0	1	YİĞİN←BC
PUSH D	RI	3	12	-	1	1	0	1	0	1	0	1	YİĞİN ←DE
PUSH H	RI	3	12	-	1	1	1	0	0	1	0	1	YİĞİN ←HL
PUSH PSW	RI	3	12	-	1	1	1	1	0	1	0	1	YİĞİN ←AF
POP B	RI	3	10	-	1	1	0	0	0	0	0	1	BC← YİĞİN
POP D	RI	3	10	-	1	1	0	1	0	0	0	1	DE← YİĞİN
POP H	RI	3	10	-	1	1	1	0	0	0	0	1	HL← YİĞİN
POP PSW	RI	3	10	-	1	1	1	1	0	0	0	1	AF← YİĞİN
XTHL	RI	5	18	-	1	1	1	0	0	0	1	1	HL<-> YİĞİN
EI	-	1	4	-	1	1	1	1	0	0	1	1	Kesmeyi yetkili kılma
DI	-	1	4	-	1	1	1	1	0	0	1	1	Kesmeyi yetkisiz kılma
NOP	-	1	4	-	0	0	0	0	0	0	0	0	İşlem yok boş geç
HLT	-	1	5	-	0	1	1	1	1	1	1	0	Mikroişlemciyi durdur
RIM	-	1	4	-	0	0	1	0	0	0	0	0	Kesme bayrağını oku
SIM	-	1	4	-	0	0	1	1	0	0	0	0	Kesme bayrağını 1 yap



## EK-B PIC16F84 ÖZEL KAYDEDİCİLERİ ve AÇIKLAMALARI

### Status Kaydedici (Adres 03H, 83H)

R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-X	R/W-X	R/W-X
IRP	RP1	RP0	T0	PD	Z	DC	C
W: yazılabilir bit			R: okunabilir bit		U: tanımlanmamış bit		
Bit No	Kaydedici Adı		İşlevi / Açıklaması				
7	Kaydedici Bank Seçme Biti (IRP)		IRP biti, PIC içerisinde bulunan dahili RAM belleğin dolaylı adreslenebilmesi amacıyla 8 bit olarak işlem görür. IRP biti, 16F8X serisinde '0' olarak tutulur.				
	Lojik '1':		Bank 2 – 3 bölgelerinin				
	Lojik '0':		Bank 0 – 1 bellek bölgelerinin seçildiğini gösterir.				
5-6	Kaydedici Bank Seçme Bitleri (RP1/RP0)		RP1 ve RP0 bitleri doğrudan adreslemede adresin yüksek değerli kısmını temsil ederler.				
	Lojik '01':		Bank 1				
	Lojik '00':		Bank 0				
4	Zaman Aşımı/Gözetleme Sayacı Taşma Biti (TO)		TO biti, besleme gerilimi uygulanması ve 'CLRWDT' ile 'SLEEP' komutlarının işlenmesi durumunda lojik '1' değerine kurulurken, 'gözetleme sayacının' en yüksek değerine ulaşması (zaman aşımı) durumunda lojik '0' değerini alır.				
	Lojik '1':		Güç verilince, CLRWDT / SLEEP komutu çalıştırılınca.				
	Lojik '0':		WDT'de taşma / zaman aşımı oluşması durumunda.				
3	Güç Azaltma Biti (PD)		Ayrıca 'RB0/INT' pinine bir sinyal uygulanmasıyla, RB portu üzerindeki değişiklik yapılmasıyla 'lojik 1' değerine kurulabilir.				
	Lojik '1':		Güç kaynağı uygulanınca ve CLRWDT komutundan sonra.				
	Lojik '0':		'Uyuma / Sleep' komutunun çalıştırılmasından sonra.				
2	Sıfır Biti (Z)		Z biti, aritmetik ve mantık işlemlerindeki sonuca göre değer alır.				
	Lojik '1':		Aritmetik / Lojik işlemlerin sonucu '0' ise.				
	Lojik '0':		Aritmetik / Lojik işlemlerin sonucu '0' değerinden farklı ise.				
1	Yardımcı Elde Biti (DC)		Toplama ve çıkarma işlemlerinde, düşük değerli dört bitlik işlemde diğer kısımlara elde veya borç olması durumunu gösterir.				
	Lojik '1':		Dördüncü bitten elde / borç oluşması durumunu gösterir.				
	Lojik '0':		Dördüncü bitte elde/borç olmadığını gösterir.				
0	Elde Biti (C)		Elde biti, toplama, çıkarma ve kaydırma işlemlerinden etkilenir.				
	Lojik '1':		En yüksek değerli bitte elde oluşması durumunu gösterir.				
	Lojik '0':		Elde değeri oluşmaması durumunu gösterir.				

**Option Kaydedici (Adres 81H)**

R/W-I	R/W-I	R/W-I	R/W-I	R/W-I	R/W-I	R/W-I	R/W-I
RPBU	INTEDG	TOCS	TOSE	PSA	PS2	PS1	PS0
W: yazılabilir bit		R: okunabilir bit		U: tanımlanmamış bit			
Bit No	Kaydedici Adı	İşlevi / Açıklaması					
7	Yetkilendirme biti (RBPU) Lojik '1': Lojik '0':	PORB Pull-up'ını yetkilendirir. Portb'nin pull-up'ı yetkisiz Portb'nin pull-up'ı yetkili					
6	Kenar seçme biti (INTEDG) Lojik '1': Lojik '0':	Kesme için kenar seçer. RB0/INT pininde yükselen kenarda kesme RB0/INT pininde düşen kenarda kesme					
5	Kaynak seçim biti (TOCS) Lojik '1': Lojik '0':	TMR0 Clock (saat) kaynağını seçer RA4/TOCKI pinindeki değişim Dahili komut saykıl clock'u (CLKOUT)					
4	Kaynak kenar seçim biti (TOSE) Lojik '1': Lojik '0':	TMR0 Kaynak kenar seçimi yapar RA4/TOCKI pininde yüksekte düşüğe geçişte arttırılır RA4/TOCKI pininde düşüğe yükseğe geçişte arttırılır					
3	Ölçekleme biti (PSA) Lojik '1': Lojik '0':	Prescaler atama biti Prescaler WDT için atanır Prescaler TMR0 için atanır					
Bit 2-0: PS2-PS0: Prescaler Oranı Seçimi (8 farklı frekans bölme değeri seçilebilir)							
Bit Değeri	TMR0 Oranı	WDT Oranı					
000	1:2	1:1					
001	1:4	1:2					
010	1:8	1:4					
011	1:16	1:8					
100	1:32	1:16					
101	1:64	1:32					
110	1:128	1:64					
111	1:256	1:128					

**INTCON Kaydedici (Adres 0BH,8BH)**

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-X
GIE	EEIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF
W: yazılabilir bit			R: okunabilir bit		U: tanımlanmamış bit		
Bit No	Kaydedici Adı		İşlevi / Açıklaması				
7	Genel Kesme Yetkilendirme Biti (GIE)		GIE kaydedicisi 7. biti tüm kesmelerin yetkilendirilmesi veya kesmelerin devre dışı bırakılması sağlar. Lojik '1': Maskelenemez tüm Kesmeler yetkilendirilir Lojik '0': Tüm Kesmeler Yetkisizlendirilir.				
6	EEPROM Yazma Tamamlama Kesme Yetkilendirme Biti (EEIE)		EEIE biti, EEPROM'a yazma işleminin sonuçlanması ile 'EE kesme tamamlandı' kesmesini yetkilendirir. IEIE ve EFCON 1 kaydedicisindeki EEIF bitlerinin aynı zamanda 'lojik 1' yapılması durumunda kesme oluşur. Lojik '1': EE Kesmesi Yetkilendirilir Lojik '0': EE Kesmesi Yetkisizlendirilir				
5	TMRO	Taşma Kesme Yetkilendirme Biti (TOIE)	TMRO sayıcısının taşması durumunda kesmeleri yetkilendiren bittir. 'TOIE' ve 'TOIE' ve 'TOIF' bitlerinin aynı anda 'lojik 1' olması durumunda kesme oluşur. Lojik '1': Kesme Yetkilendirilir Lojik '0': Kesme Yetkisizlendirilir.				
4	INTE Harici Kesme Yetkilendirme Biti (INTE)		'INTE' biti, RBO/INT pininden gelen harici kesmeleri yetkilendirir. 'INTE' ve 'INTF' bitlerinin aynı anda 'lojik 1' yapılması durumunda kesme oluşur. Lojik '1': Harici Kesmeler Yetkilendirilir Lojik '0': Harici Kesmeler Yetkisizlendirilir				
3	RB	Port Değişikliği Kesme Yetkilendirme Biti (RBIE)	RBIE biti, B portundaki 4,5,6 ve 7 no'lu pinlerde değişim olması durumunda kesmeler yetkilendirir. 'RBIE' ve 'RBIF' kaydedicilerinin 'lojik 1' olması durumunda kesme oluşur. Lojik '1': Durum Değişikliklerinde Kesmeleri Yetkilendirilir Lojik '0': Durum Değişikliklerinde Kesmeleri Yetkisizlendirir				
2	TMRO Tasma Kesme Bayrağı Biti (TOIF)		TOIF biti, TMRO sayıcısında taşma oluşması durumunda kesmeyi kontrol eden programların işlenmesi sırasında bir kesmenin hissedilebilmesi / fark edilebilmesi için bit 'lojik 0' yapılmaktadır. (FF) <sub>16</sub> durumunda (00) <sub>16</sub> durumuna değiştiğinde oluşur. Lojik '1': Taşma Oluşmaz Lojik '0': Taşma Oluşmaz				
1	INT Harici Kesme Bayrağı Biti (INTF)		Harici kesme oluşur. RBO/INT pini üzerinde yüksele veya düşen kenar hissedildiğinde INTF biti 'lojik 1' yapılır. Lojik '1': Kesme Oluşur Lojik '0': Kesme Oluşmaz				
0	RB	Port Değişikliği Kesme Bayrağı Biti (RBIF)	RBIF biti, B portunda 4, 5, 6, 7 no'lu pinlerde değişiklik olması durumunu belirtir. Daha sonra olabilecek kesmeleri fark edilecek amacıyla kesme alt programlarında 'lojik 0' yapılmaktadır. Lojik '1': Belirtilen pinlerden en az birisinin durum değiştirme durumunda Lojik '0': Herhangi bir pinde değişiklik olmaması durumunda.				

**EECON Kaydedici (Adres 88H,89H)**

			R/W	R/W	R/W	R/S	R/S
----	----	----	EEIF	WREER	WREN	WR	RD
W: yazılabilir bit			R: okunabilir bit			U: tanımlanmamış bit	
Bit No	Kaydedici Adı		İşlevi / Açıklaması				
4	EEPROM Yazma Biti (EEIF)		EEPROM'a yazma işleminin bittiğini gösterir.				
		Lojik '1':	Yazma bitti veya sonlandırıldı				
		Lojik '0':	Yazma bitmedi veya başlamadı				
3	EEPROM Hata Bayrak Biti (WREER)		EEPROM'a veri yazılırken hata durumunu gösterir				
		Lojik '1':	Hata oluştuğu durumu gösterir				
		Lojik '0':	Hata oluşmadığı durumu gösterir				
2	EEPROM yazma izin biti (WREN)		EEPROM'a veri yazılmasına izin verir / yetkilendirir				
		Lojik '1':	Yazma izni verilir				
		Lojik '0':	Yazma izni verilmez				
1	Yazma Kontrol Biti (WR)		EEADR kaydedicisinde bulunan adrese EEDATA kaydedicisinin içeriğinin kopyalanmasını sağlar				
		Lojik '1':	Yazma işlemi başlar				
		Lojik '0':	Yazma işlemine başlanmaz				
0	Okuma Kontrol biti (RD)		EEADR Kaydedicisinde tanımlı olan adresten EEDATA kaydedicisine veri iletimini kontrol eder.				
		Lojik '1':	Okuma işlemi başlar				
		Lojik '0':	Okuma işlemine başlanmaz				

## EK-C PIC16F84 KOMUTLARI ve AÇIKLAMALARI

Tüm komutlar 14 bitlik opcode sahiptir ve tek kelimeliktir. Komutların opcode ve Q aktiviterinde kullanılan terimlerin açıklamaları şu şekildedir:

- f: 7 bit dosya kaydedici adresidir.  $0 \leq f \leq 127$
- d: Sonucun aktarılacağı hedef kaydediciyi belirler. Alacağı iki değer vardır:  
d=0 ise hedef W kaydedicisi,  
d=1 ise hedef f kaydedicisidir.
- k: GOTO ve CALL çağırma komutları için bir adres gösterir ve 11 bitlidir ve k değeri  $0 \leq k \leq 2047$  aralığındadır. Diğer komutlarda 8 bitlidir ve k değeri  $0 \leq k \leq 255$  aralığında sayısal bir değeri temsil eder.

Q Aktiviteleri: Komutların çoğunda kullanılan Q aktiviteleri aşağıda verilmiştir. Q aktiviteleri bu düzenlerde olmayan komutların Q aktiviteleri komut anlatılırken verilmiştir.

	Saykıl	Q1	Q2	Q3	Q4
LW k	1	Kodu Çöz	k'yı Oku	Veriyi İşle	W'ye Yaz
f,d	1	Kodu Çöz	f'yi Oku	Veriyi İşle	Hedefe Yaz
BCF-BSF-CLRF	1	Kodu Çöz	f'yi Oku	Veriyi İşle	f'ye Yaz

**ADDLW:** W kaydedici içeriği ile 8-bit gerçek değer k'yı toplar.

W'nin içeriği ile k'yı toplar	opcode	11	111x	kkkk	kkkk
İfade: [label] ADDLW k	STATUS: C,DC,Z				
İşlem: (W) + (k) → (W)	Cycles: 1				

Örnek: ADDLW 0x15

Komuttan önce

W=0x10

Komuttan sonra  $W=0x25$

ADDWF: W kaydedicisinin içeriği ile f kaydedicisi içeriğini toplar.

W ile f kaydedici içeriklerini toplar	opcode	00	0111	dfff	ffff
İfade: [label] ADDWF f,d	STATUS: C,DC,Z				
İşlem: (W) + (f) → (hedef)	Cycles: 1				

Örnek: ADDWF FSR,0

Komuttan önce  $W=0x17$  FSR=0xC2

Komuttan sonra  $W=0xD9$  FSR=0xC2

ANDLW: W kaydedici içeriği ile 8-bit gerçek değer k'yı AND'ler. Sonucu W kaydedicisine kaydeder.

W ile k'yı AND'ler	opcode	11	1001	kkkk	kkkk
İfade: [label] ANDLW k	STATUS: Z				
İşlem: (W).AND.(k) → (W)	Cycles: 1				

Örnek: ADDLW 0x5F

Komuttan önce  $W=0xA3$

Komuttan sonra  $W=0x03$

ANDWF: W kaydedici içeriği ile f kaydedici içeriğini AND'ler.

W ile f'yi AND'ler	opcode	00	0101	dfff	ffff
İfade: [label] ANDWF f,d	STATUS: Z				
İşlem: (W).AND.(f) → (hedef)	Cycles: 1				

Örnek: ADDWF FSR,1

Komuttan önce Komuttan sonra

$W=0x17$   $W=0x17$

FSR=0xC2 FSR=0x02

BCF: f kaydedicisindeki b bitini temizler

f kaydedicisindeki b bitini temizler	opcode	01	00bb	bfff	ffff
İfade: [label] BCF f,b	STATUS: yok				
İşlem: 0 → (f<b>)	Cycles: 1				

Örnek: BCF FLAG\_REG, 7

Komuttan önce            FLAG\_REG=0xC7  
 Komuttan sonra            FLAG\_REG=0x47

BSF: f kaydedicisindeki b bitini set eder.

f kaydedicisindeki b bitini set eder	opcode	01	01bb	bfff	ffff
İfade: [label] BSF f,b	STATUS: yok				
İşlem: 1 → (f<b>)	Cycles: 1				

Örnek:            BSF FLAG\_REG, 7

Komuttan önce            FLAG\_REG=0x0A  
 Komuttan sonra            FLAG\_REG=0x8A

BTFSC: Şayet f kaydedicisindeki b biti 1 ise bir sonraki komut çalıştırılır aksi halde bu komut atlanır ve bu komut yerine bir NOP komutu icra edilir.

Bit test eder	opcode	01	10bb	bfff	ffff
İfade: [label] BTFSC f,b	STATUS: yok				
İşlem: atla (f<b>)=0 ise	Cycles: 1(2:atlama gerçekleşirse)				

Q Aktivitesi:		Q1	Q2	Q3	Q4
	1. saykıl	Kodu Çöz	f'yi Oku	Veriyi İşle	İşlem Yok
	2. saykıl	İşlem Yok	İşlem Yok	İşlem Yok	İşlem Yok

Örnek:

Başla            BTFSC FLAG, 1  
 Yanlış            GOTO PROCESS\_CODE  
 Doğru            .....

Komuttan önce            Komuttan sonra  
 PC=adres başla            Eğer FLAG<1>=0 PC=adres “doğru”  
     Eğer FLAG<1>=1 PC=adres “yanlış”

BTFSS: Şayet f kaydedicisindeki b biti 0 ise bir sonraki komut çalıştırılır aksi halde bu komut atlanır ve bu komut yerine bir NOP komutu icra edilir.

bit test eder	opcode	01	11bb	bfff	ffff
İfade: [label] BTFSS f,b	STATUS: yok				
İşlem: atla (f<b>)=1 ise	Cycles: 1(2:atlama gerçekleşirse)				

Q Aktivitesi:		Q1	Q2	Q3	Q4
	1. saykıl	Kodu Çöz	f'yi Oku	Veriyi İşle	İşlem Yok
	2. saykıl	İşlem Yok	İşlem Yok	İşlem Yok	İşlem Yok

Örnek:

Başla            BTFSS FLAG, 1

Yanlış           GOTO PROCESS\_CODE

Doğru            .

Komuttan önce

Komuttan sonra

PC=adres başla

Eğer FLAG<1>=0 PC=adres “doğru”

Eğer FLAG<1>=1 PC=adres “yanlış”

CALL: Altprograma dallan. İlk olarak yığına (PC+1) adresi atılır. Dallanma adresinin ilk 11-biti PC'ye yüksek bitlerini ise PCLATH'a kaydeder.

Alt programa dallanmayı sağlar.	opcode	10	0kkk	kkkk	kkkk
İfade: [label] CALL k	STATUS: yok				
İşlem: (PC)+1→TOS - k→PC<10:0>, (PCLATH<4:3>)→PC<12:11>	Cycles: 2				

Q Aktivitesi:		Q1	Q2	Q3	Q4
	1. saykıl	Kodu Çöz	k'yı Oku PC Yığına at	Veriyi İşle	PC Geri Al
	2. saykıl	İşlem Yok	İşlem Yok	İşlem Yok	İşlem Yok

Örnek:

HERE CALL THERE

Komuttan önce    PC = adres here

Komuttan sonra    PC = adres THERE        TOS = adres HERE +1

CLRF: f kaydedicisinin içeriği temizlenir ve Z biti set edilir.

f kaydedicisini temizler.	opcode	00	0001	1fff	ffff
İfade: [label] CLRF f	STATUS: Z				
İşlem: 00→(f) / 1→Z	Cycles: 1				



Örnek: CLRFLAG\_REG  
 Komuttan önce FLAG\_REG=0x5A  
 Komuttan sonra FLAG\_REG=0x00 Z=1

CLRW: W kaydedicisinin içeriği temizlenir ve Z biti set edilir.

W kaydedicisini temizler.	opcode	00	0001	0xxx	xxxx
İfade: [label] CLRW	STATUS: Z				
İşlem: 00→(W) / 1→Z	Cycles: 1				

Örnek: CLRW  
 Komuttan önce W=0x5A  
 Komuttan sonra W=0x00 Z=1

CLRWDW: Watchdog Timer'ı resetler. Aynı zamanda WDT nin prescaler değeri resetlenir. Status kaydedicisinin 3. ve 4. bitlerini set eder.

Watchdog Timer'ı temizler	opcode	00	0000	0110	0100
İfade: [label] CLRWDW	STATUS: TO, PD				
İşlem: 00→(WDT) / 0→ WDT prescaler; 1→TO ; 1→PD /	Cycles: 1				

Q Aktivitesi:	1. saykıl	Q1	Q2	Q3	Q4
		Kodu Çöz	İşlem Yok	Veriyi İşle	WDT Temizle

Örnek: CLRWDW  
 Komuttan önce Komuttan sonra  
 WDT=? WDT Counter=0x00  
 WDT Prescaler=0  
 TO=1 ; PD=1

COMF: f kaydedicisinin içeriğinin tersini (complement) alır.

f kaydedicisinin içeriğinin tersini alır	opcode	00	1001	dfff	ffff
İfade: [label] COMF f,d	STATUS: Z				
İşlem: (f)→(hedef)	Cycles: 1				

Örnek: COMF REG1, 0

Komuttan önce            REG1 = 0x13  
 Komuttan sonra            REG1 = 0x13    W = 0xEC

DECF: f kaydedicisinin içeriğini bir azaltır.

f kaydedicisinin içeriğini 1 azaltır	opcode	00	0011	dfff	ffff
İfade: [ <i>label</i> ] DECF f,d	STATUS: Z				
İşlem: (f)-1→(hedef)	Cycles: 1				

Örnek:            DECF CNT,1  
 Komuttan önce            CNT = 0x01    Z=0  
 Komuttan sonra            CNT =0x01    Z=1

DECFSZ: f kaydedicisinin içeriğini azaltır.

f-1,Sonuç 0 ise bir komut atlar	opcode	00	1011	dfff	ffff
İfade: [ <i>label</i> ] DECFSZ f,d	STATUS: yok				
İşlem: (f)-1→(hedef)	Cycles: 1(2:atlama gerçekleşirse)				

Q Aktivitesi:		Q1	Q2	Q3	Q4
	1. saykıl	Kodu Çöz	f'yi Oku	Veriyi İşle	Hedefe Yaz
	2. saykıl	İşlem Yok	İşlem Yok	İşlem Yok	İşlem Yok

Örnek: HERE DECFSZ CNT,1

GOTO LOOP

DEVAM

\*

Komuttan önce            Komuttan sonra  
 PC = adres HERE            CNT =CNT-1  
                                  Şayet CNT=0, PC= adres DEVAM  
                                  Şayet CNT≠0, PC=adres HERE+1

GOTO: Koşulsuz dallanmadır. K'nın 11-bitlik kısmı PC<10:0>'a diğer kısmı ise PCLATH <4:3>'a yüklenir. GOTO komutu 2 saykılta işlenir.

Koşulsuz Dallanma komutu	opcode	10	1kkk	kkkk	kkkk
İfade: [ <i>label</i> ] GOTO k	STATUS:				

İşlem: $k \rightarrow PC <10:0> /$ $PCLATH <4:3> \rightarrow PC <12:11>$	Cycles: 2
---	-----------

Q Aktivitesi:		Q1	Q2	Q3	Q4
	1. saykıl	Kodu Çöz	k'yi Oku	Veriyi İşle	PCye Yaz
	2. saykıl	İşlem Yok	İşlem Yok	İşlem Yok	İşlem Yok

Örnek: GOTO HERE

Komuttan sonra PC = adres THERE

INCF: f kaydedicisinin içeriğini bir arttırır. .

f kaydedicisinin içeriği 1 arttırır	opcode	00	1010	dfff	ffff
İfade: $[label] INCF f,d$	STATUS: Z				
İşlem: $(f)+1 \rightarrow (hedef)$	Cycles: 1				

Örnek: INCF CNT,1

Komuttan önce CNT=0xFF Z=0

Komuttan sonra CNT=0x00 Z=1

INCFSZ: f kaydedicisinin içeriğini bir arttırır. Sonuç lojik "1" ise bir sonraki komutu çalıştırır. Aksi halde bir komut atlanarak program çalışmasına devam eder.

f+1, Sonuç 0 ise komut atlar.	opcode	00	1111	dfff	ffff
İfade: $[label] INCFSZ f,d$	STATUS:yok				
İşlem: $(f) + 1 \rightarrow (hedef)$	Cycles: 1(2:atlama gerçekleşirse)				

Q Aktivitesi:		Q1	Q2	Q3	Q4
	1. saykıl	Kodu Çöz	f'yi Oku	Veriyi İşle	Hedefe Yaz
	2. saykıl	İşlem Yok	İşlem Yok	İşlem Yok	İşlem Yok

Örnek: HERE INCFSZ CNT,1

GOTO LOOP

DEVAM \*

Komuttan Önce Komuttan Sonra

PC=adres HERE CNT=CNT+1

Şayet CNT=0, PC=adres DEVAM

Şayet CNT≠0, PC=adres HERE+1

IORLW: W kaydedici içeriği ile 8 bit gerçek değer k OR'lanır. Sonuç W'ye kaydedilir.

W ile k OR işlemine tabi tutulur.	opcode	11	1000	kkkk	kkkk
İfade: <i>[label]</i> IORLW k	STATUS: Z				
İşlem: (W).OR.k → (W)	Cycles: 1				

Örnek: IORLW 0x35

Komuttan önce W=0x9A

Komuttan sonra W=0xBF Z=1

IORWF: W ile f kaydedicilerinin içerikleri OR'lanır.

W ile f kaydedicisi OR'lanır	opcode	00	0100	dfff	ffff
İfade: <i>[label]</i> IORWF f,d	STATUS: Z				
İşlem: (W).OR.(f) →(hedef)	Cycles: 1				

Örnek: IORWF RESULT, 0

Komuttan önce RESULT=0x13 W=0x91

Komuttan sonra RESULT=0x13 W=0x93 Z=1

MOVF: d'nin durumuna göre f kaydedicisinin içeriğini hedefe yazar.

f kaydedicisinin içeriğini yükle	opcode	00	1000	dfff	ffff
İfade: <i>[label]</i> MOVF f,d	STATUS: Z				
İşlem: (f)→(hedef)	Cycles: 1				

Q Aktivitesi:	1. saykıl	Q1	Q2	Q3	Q4
		Kodu Çöz	f'yi Oku	Veriyi İşle	Hedefe Yaz

Örnek: MOVF FSR, 0

Komuttan sonra W=FSR Z=1

MOVLW: W kaydediciye 8-bitlik sayı k aktarılır.

k sayısını W'ye aktarır.	opcode	11	00xx	kkkk	kkkk
İfade: <i>[label]</i> MOVLW k	STATUS: yok				
İşlem: (k)→(W)	Cycles: 1				

Örnek:           MOVLW0x5A  
                   Komuttan Sonra       W=0x5A

MOVWF: W kaydedicisi içeriği f kaydedicisine aktarılır.

W'yi f'ye aktarır.	opcode	00	0000	1fff	ffff
İfade: <i>[label]</i> MOVWF f	STATUS: yok				
İşlem: (W)→(f)	Cycles: 1				

Q Aktivitesi:	1. saykıl	Q1	Q2	Q3	Q4
		Kodu Çöz	f'yi Oku	Veriyi İşle	f'ye Yaz

Örnek:           MOVWF OPTION\_REG  
                   Komuttan Önce       OPTION\_REG=0xFF       W=0x4F  
                   Komuttan Sonra    OPTION\_REG=0x4F       W=0x4F

NOP: İşlem yok

işlem yok	opcode	00	0000	0xx0	0000
İfade: <i>[label]</i> NOP	STATUS:				
İşlem: işlem yok	Cycles: 1				

Q Aktivitesi:	1. saykıl	Q1	Q2	Q3	Q4
		İşlem Yok	İşlem Yok	İşlem Yok	İşlem Yok

RETFIE: Kesme alt programından geri dönme komutu. Yığına atılan veriler geri alınır ve yığının en üstündeki(TOS) veri PC'a yüklenir. Global Kesme Yetkilendirme (GIE) set edilir. 2 saykılta işlenen bir koddur.

Alt rutinden geri dönmeyi sağlar	opcode	00	0000	0000	1001
İfade: <i>[label]</i> RETFIE	STATUS: yok				
İşlem: TOS→ PC / 1→GIE	Cycles: 2				

Q Aktivitesi:		Q1	Q2	Q3	Q4
	1. saykıl	Kodu Çöz	İşlem Yok	GIeyi set et	Yığıcı Geri Al
	2. saykıl	İşlem Yok	İşlem Yok	İşlem Yok	İşlem Yok

RETLW: Alt programdan geri dönmeyi sağlar. Yığıcı atılan bilgileri geri alınır ve TOS, PC' ye geri yüklenir. k değeri W kaydedicisine yüklenir.

Alt programdan geri dönmeyi sağlar.	opcode	11	01xx	kkkk	kkkk
İfade: [label] RETLW k	STATUS: yok				
İşlem: k → (W) / TOS → PC	Cycles: 2				

Q Aktivitesi:		Q1	Q2	Q3	Q4
	1. saykıl	Kodu Çöz	k'yı oku	İşlem Yok	W'ye yaz Yığıcı Geri Al
	2. saykıl	İşlem Yok	İşlem Yok	İşlem Yok	İşlem Yok

RETURN: Alt programdan geri dönmeyi sağlar. Yığıcı atılan bilgileri geri alınır ve TOS, PC' ye geri yüklenir. 2 saykıl işlenen bir koddur.

Alt programdan geri dönmeyi sağlar.	opcode	00	0000	0000	1000
İfade: [label] RETURN	STATUS: yok				
İşlem: TOS → PC	Cycles: 2				

Q Aktivitesi:		Q1	Q2	Q3	Q4
	1. saykıl	Kodu Çöz	İşlem Yok	İşlem Yok	Yığıcı Geri Al
	2. saykıl	İşlem Yok	İşlem Yok	İşlem Yok	İşlem Yok

RLF: f kaydedicisinin içeriği C biti kullanılarak sola doğru kaydırma işlemiyle bir döngü sağlanır.

f içeriği C bitini kullanarak sola kayar.	opcode	00	1101	dfff	ffff
İfade: [label] RLF f,d	STATUS: C				
İşlem: -	Cycles: 1				

Örnek: RLF REG1,0

Komuttan önce	Komuttan sonra
REG1=1110 0110	REG1=1110 0110
W=1000 1100	W=1100 1100
C=1	C=1

RRF: f kaydedicisinin içeriği C biti kullanılarak sağa doğru kaydırma işlemiyle bir döngü sağlanır.

f içeriği C bitini kullanarak sağa kayar.	opcode	00	1100	dfff	ffff
İfade: [label] RRF f,d	STATUS: C				
İşlem: -	Cycles: 1				

Örnek: RRF REG1,0

Komuttan önce	Komuttan sonra
REG1=1110 0110	REG1=1110 0110
W=1000 1100	W=0111 0011
C=0	C=0

SLEEP: STATUS kaydedicisinin Power-down biti temizlenir. Bunun yanında Time-Out biti set edilir. Watchdog Timer ve ona ait prescaler temizlenir. İşlemci osilatörü durdurarak SLEEP moduna geçer.

Mikrodenetleyiciyi uyku moduna geçirir	opcode	00	0000	0110	0011
İfade: [label] Sleep	STATUS: TO, PD				
İşlem: 00 → WDT 0 → WDT prescaler; 1 → TO ; 0 → PD	Cycles: 1				

Q Aktivitesi:	1. saykıl	Q1	Q2	Q3	Q4
		Kodu Çöz	İşlem Yok	İşlem Yok	SLEEP Modu

SUBLW: 8-bitlik k sayısından W kaydedici (2'nin komplementi metodu kullanılarak) çıkarılır. Sonuç W'ye kaydedilir.

k sayısından W'yi çıkarır	opcode	11	110x	kkkk	kkkk
İfade: [label] SUBLW K	STATUS: C,DC,Z				
İşlem: k - (W) → (W)	Cycles: 1				

Örnek:	SUBLW 0x02				
Örnek1:	Komuttan önce	W=1	C=?	Z=?	
	Komuttan sonra	W=1	C=1; sonuç pozitif	Z=0	
Örnek2:	Komuttan önce	W=2	C=?	Z=?	
	Komuttan sonra	W=0	C=1; sonuç sıfır	Z=0	
Örnek3:	Komuttan önce	W=3	C=?	Z=?	
	Komuttan sonra	W=0xFF	C=0; sonuç negatif	Z=0	

SUBWF: f kaydedici içeriğinden W kaydedici içeriği çıkarılır. d'nin değerine göre kaydedicilerden birine sonuç aktarılır.

f kaydedicisinden W'yi çıkarır.	opcode	00	0010	dfff	ffff
İfade: [label] SUBWF f,d	STATUS: C,DC,Z				
İşlem: (f) - (W) → (hedef)	Cycles: 1				

Örnek: SUBWF REG1, 1

Örnek 1:	Komuttan önce	REG1=3	W=2	C=?	Z=?
	Komuttan sonra	REG1=3	W=2	C=1; sonuç pozitif	Z=0
Örnek 2:	Komuttan önce	REG1=2	W=2	C=?	Z=?
	Komuttan sonra	REG1=2	W=2	C=1; sonuç sıfır	Z=1
Örnek 3:	Komuttan önce	REG1=1	W=2	C=?	Z=?
	Komuttan sonra	REG1=1	W=2	C=0; sonuç negatif	Z=0

SWAPF: f kaydedicisinin Nibble'lerini değiştirir.

f kaydedicisinin Nibble'lerini değiştirir.	opcode	00	1110	dfff	ffff
İfade: [label] SWAPF f,d	STATUS: yok				
İşlem: (f<3:0>) → (hedef <7:4>) (f<7:4>) → (hedef <3:0>)	Cycles: 1				

XORLW: W kaydedici içeriği ile 8 bit gerçek değer k XOR'lanır. Sonuç W'ye kaydedilir.

W ile k XOR işlemine tabi tutulur.	opcode	11	1010	kkkk	kkkk
İfade: [label] XORLW k	STATUS: Z				
İşlem: (W).XOR.k → (W)	Cycles: 1				



Örnek: XORLW 0xAF  
 Komuttan önce Komuttan sonra  
 W=0xB5 W=0x1A

XORWF: W ile f kaydedicilerinin içerikleri XOR'lanır.

W ile f kaydedicisi XOR'lanır	opcode	00	0110	dfff	ffff
İfade: [ <i>label</i> ] XORWF f,d	STATUS: Z				
İşlem: (W).XOR.f →(hedef)	Cycles: 1				

Örnek: XORWF REG, 1  
 Komuttan önce REG=0xAF W=0xB5  
 Komuttan sonra REG =0x1A W=0xB5

## ÖZGEÇMİŞ

Aytaç KAYA 1975 yılında Sakarya’da doğdu. İlköğrenimini Sakarya’da 1986 ve ortaöğrenimi İstanbul 1988’de tamamladı. Lise öğrenimini Avcılar-İstanbul Avcılar Endüstri Meslek Lisesinde 1992’de tamamladı. Aynı yıl girdiği Gazi Üniversitesi Teknik Eğitim Fakültesi Elektronik-Bilgisayar Öğretmenliği bölümünden 1997’de mezun oldu. 1998 yılında Avcılar –İstanbul Avcılar Endüstri Meslek Lisesinde tek dönem öğretmenlik yaptı. Aynı yılın dönem bitiminde Sakarya Üniversitesi Bilgişlem Merkezinde Yazılım Uzmanı olarak göreve başladı. 2001 yılında Sakarya Üniversitesi Eğitim Fakültesinde öğretim görevlisi kadrosuyla çalışmaya başladı ve halen aynı okulda öğretim görevlisi olarak çalışmaktadır.