T.C.
FIRAT ÜNİVERSİTESİ
TEKNOLOJİ FAKÜLTESİ
YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ

YMT216 MİKROİŞLEMCİLER VE PROGRAMLAMA

(Ders Notları)

İbrahim TÜRKOĞLU

ELAZIĞ – 2012

İÇİNDEKİLER

BÖLÜM 1. GİRİŞ

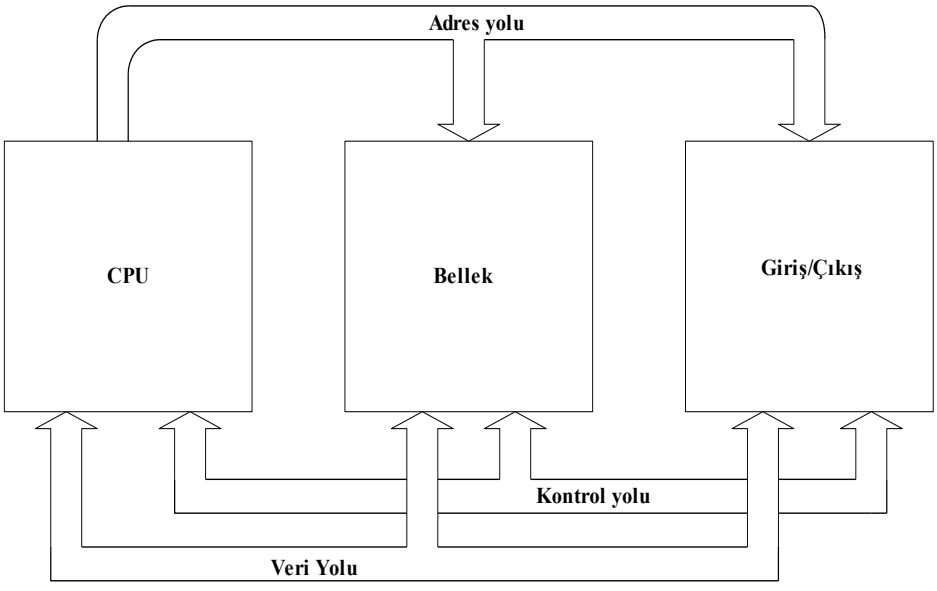
Günümüzde hızla gelişen teknoloji bilgisayarla kontrol edilen cihazları bizlere çok yaklaştırdı. Öyle ki günlük hayatımızda sıkça kullandığımız bir çok elektronik cihaz (cep telefonu, faks, oyuncaklar, elektrikli süpürge, bulaşık makinası, vs.) artık çok küçük sayısal bilgisayarlarla (mikro denetleyiciler) işlev kazandırılabilmektedir. Benzer işler, ilk zamanlarda mikroişlemci tabanlı bilgisayar kartları ile yapılabilmekteydi. Mikroişlemci ile bir cihazı kontrol etme işlemi Giriş/Çıkış ve hafıza elemanı gibi ek birimlere ihtiyaç duyar. Böylesi bir tasarım kolay olmamakla birlikte, maliyet ve programlama açısından da dezavantajlara sahiptir. İşte mikrodenetleyiciler bu sorunları ortadan kaldırmak ve bir çok fonksiyonu tek bir entegrede toplamak üzere tasarlanmış olup, günümüzde hemen hemen bir çok elektronik cihazda farklı tipleri bulunmaktadır.

Mikroişlemci, saklı bir komut dizisini ardışıl olarak yerine getirerek veri kabul edebilen ve bunları işleyebilen sayısal bir elektronik eleman olarak tanımlanabilir. Mikroişlemci temelde mantık kapıları, flip-floplar, sayıcı ve saklayıcılar gibi standart sayısal devrelerden oluşur.

Genel olarak bilgisayar ile iki şekilde ilgilenilir :

1. Yazılım (Software) : Bilgisayarın fiziksel parçalarını işler hale getiren bileşenlerdir.
2. Donanım (Hardware) : Bilgisayarı oluşturan fiziksel parçaların tümüdür.

Her ikisi de birbirinin tamamlayıcısıdır.Birisi olmazsa diğeri de olmaz.Sistem öncelikli olarak tasarlanırken önce sistemi meydana getirecek elemanlar ,yani donanım parçaları göz önüne alınır.Daha sonra yazılım bu yapıya bakılarak yazılır.Yazılım,donanımın hangi yönteme göre nasıl çalışacağını gösteren bir sanal uygulamadır. Hangi zamanda hangi elemanın devreye girerek üzerindeki bilgiyi işlemesini sağlamaktadır. Basit bir bilgisayarın ana elemanları **Şekil 1.1.'de** görülmektedir. Tüm sayısal bilgisayarlar şekilde gösterilen elemanlara sahiptirler. Bunların dışındaki eleman ya da cihazlar seçimliktir.



Şekil 1.1: Genel Bilgisayar yapısı

Bilgisayarı oluşturan bu sistemdeki elamanlar; mikroişlemci(CPU), bellek ve giriş/çıkış(G/Ç) birimleridir. Mikroişlemcinin işleyeceği komutlar ve veriler geçici veya kalıcı belleklerde tutulmaktadır. Bilgiyi oluşturan komut ve veriler bellekte karmaşık veya farklı alanlarda tutulabilir.Yazan kişinin karakterini veya seçtiği yolu gösteren çeşitli algoritmalardan meydana gelen program işlemciyi kullanarak verilerin işlenmesini sağlar.Bilginin işlenmesi sırasında ortaya çıkabilecek ara değerler ,en sonunda sonuçlar bellekte bir yerde depolanmak zorundadır.Bütün bu yapılan işlemler bir hesaba dayanmaktadır.Bilgisayarın bilgiyi işlemedeki ana karar vericisi sistemin kalbi sayılan mikroişlemcidir.CPU tarafından gerçekleştirilen iki temel işlem vardır.Birincisi komutların yorumlanarak doğru bir sırada gerçekleşmesini sağlayan kontrol işlevi,diğeri toplama,çıkarma vb özel matematik ve mantık işlemlerinin gerçekleştirilmesini sağlayan icra işlevidir.

Bilgisayarda çalıştırılan yazılımlar kendi aralarında ikiye ayrılır.Bunlar, programcı tarafından yüksek düzeyde yazılan programlardır ki insanlar tarafından anlaşılabilir düzeydedir ve bu yazılan programların makine tarafından anlaşılmasını sağlayan bağdaştırıcı (interface) yazılımlardır ki işletim sistemi(OS) olarak anılırlar.Mikroişlemci mantıksal 0 ve 1 esasına göre çalıştığından,verilen komutların da bu esasa dayanması gerekmektedir.Kısaca sayısal bilgisayarların kullandığı doğal dile makine dili denir.Programcı tarafından yüksek düzeyde yazılan programlar ancak yine insanlar tarafından anlaşılabilir.Bu programların makine tarafından anlaşılabilmesi için derleyici,yorumlayıcı ve assembler gibi aracı programların kullanılması gerekir. Demek ki, yazılım denildiğinde akla, işletim sistemi, üst düzey diller vasıtasıyla yazılan çeşitli uygulama programları gelir. Bu diller;

- Yüksek seviyeli diller
- Orta seviyeli diller
- Düşük seviyeli diller

olmak üzere üç sınıfa ayrılabilir.Bu yüksek, orta, düşük kelimelerinin anlamı donanımın yazılıma ne kadar yakın olduğunu gösterir.

Yüksek seviyeli dillerin kontrol sistemlerinde kullanımı zordur. Yüksek seviyeli bir dilde yazılan program derleyici tarafından derlendiğinde bilgisayar bunu düşük seviyeli dile (makina diline) çevirerek anlar. Orta seviyeli dillerin (assembly) kontrol sistemlerinde kullanımı uygundur.

Assembly dilini kullanırken donanımı bilmemiz zorunludur. Örneğin Intel 8085 ve Motorola 6800 mikroişlemcilerinin assembly dilleri farklıdır. Çünkü donanımları farklıdır. Orta seviyeli diller

makina dilinde, yani ikili sayı sistemi ile program yazma zor ve zahmetli bir iştir. Bunun için makina dilinin komutlar şeklinde verilmesini sağlayan assembly diller geliştirilmiştir. Assembly dilinde program yazmak makina diline göre daha kolay ve anlaşılırdır. Fakat fazla miktarda komut içerir. Bunun için anlama ve kullanımı belli bir zaman alır.

Assembly makinaya yönelik dillerdir. Programcı kullandığı bilgisayarın donanımını ve adresleme tekniklerini çok iyi bilmelidir. Assembly programları standart değildir. Aynı model olmayan her mikroişlemcinin kendine özgü assembly dili vardır**.** Programcı bu dille makinayla en basit şekilde iletişim kurar. Assembly dilinde yazılan her program bellekte saklanırken veya işlenirken 0 veya 1'ler formuna çevrilmeye gerek duyar. Bu çevirme işi programcı tarafından üretici firmanın databook kitabına bakılarak elle veya bir assembler (Assembly derleyicisi) yardımıyla yapılır.

Tek tek komut kodu karşılığına bakılarak ikili komut kodları bulunuyorsa ve eğer program çok uzun veya tekrarlamalı ise ,kaynak programı amaç programa çevirmek çok zor ve hata yapma payı yüksek olacaktır. Bu gibi durumlarda iyi bir assembler programı kullanılmalıdır.

Bazen programcılar Assembly dili ile assembleri karıştırmaktadırlar. Assembly dili, konuşma dilinde emir şeklindeki cümleden özenle seçilerek alınmış ve sayısı genelde üç en fazla dört olabilen harflerden meydana gelen ve bir komut anlam ifade eden hatırlatıcıları içerir.

Assembly dilinde program yazmak makine dilinde yazmaktan daha kolay ve takibi daha basittir**.** Fakat bu programın belleğe konulmadan önce makine diline çevrilmesi gereklidir,işte bu işi assembler denilen (bir nevi paket programda denilen) çevirici program yapar.Bu çevirme işlemine kaynak programın amaç programa çevrilmesi denir.

Assembly dilinde yazılmış bir programın amaç programa çevirmede en çok kullanılan yöntem elle yapılan işlemdir.Bu yöntemde her satırdaki hatırlatıcıya karşılık gelen kodlar üretici firma tarafından yayınlanan databook'a bakılarak bulunur. Böylece amaç program bulunmuş olur.

Assembly Dilinin Dezavantajları

- Assembly dilinde bir program yazmak için üzerinde çalışılan bilgisayarın özellikleri hakkında detaylı bilgi sahibi olunmalıdır. Mesela bunlar,bilgisayar mikroişlemcisinde bulunan kaydediciler ve sayısı,komut kümesi ve adresleme türleri gibi değişik özelliklerdir.
- Assembly dilinin diğer bir mahsuru elastiki olmamasıdır. Değişik firmalarca üretilen her mikroişlemcinin kendisine has bir programlama dili olmasıdır.Bundan dolayı bir mikroişlemci için yazılan bir assembly dilindeki program diğer bir mikroişlemcide çalışmayabilir.

Assembly Dilinin Avantajları

- Assembly dilinde program yazanlar,donanımın çalışmasını çok iyi anlamak ve ona göre iyi programlar geliştirmek zorunda olduklarından kendilerine birçok kazanımlar sağlarlar.Yüksek düzeyli dillerde program yazarken bilgisayar donanımının görünmeyen bazı yanlarına assembly dilinde sahip olunur.
- Assembly dilinde yazılan programlar yüksek düzeyli dillerle yazılan programlara nazaran daha hızlı ve küçük boyutludur. Assembly dili,program büyüklüğünde ve çalışma hızında ideal optimizasyon sağlar.

Düşük seviyeli diller ise, makina dilleridir. Yine makinaya özgü bir dildir. Bu dilde programlama çok zor, hata yapma oranı çok yüksek ve programı kontrol etme imkanı nerede ise yoktur.
Assembly ve makina diline uygun uygulamalar :

- Hesaplamalardan daha çok giriş/çıkış gerektiren uygulamalar
- Gerçek zaman denetimi ve uygulamaları
- Fazla veri işlemesi gerekmeyen uygulamalar
- Hızlılık istene uygulamalar

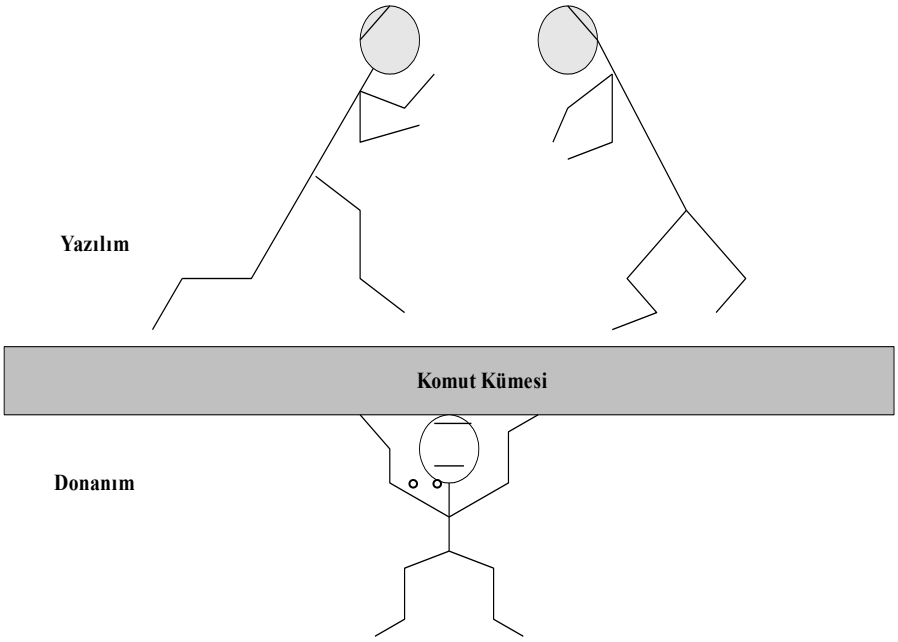
BÖLÜM 2. MİKROİŞLEMCİ VE MİKROBİLGİSAYARLAR

2.1. BİLGİSAYAR MİMARİSİ

Bilgisayar mimarisi, komut kümesinin,donanım elamanlarının ve sistem organizasyonunun dahil olduğu bir bilgisayarın tasarımıdır. Mimari iki farklı yaklaşımla tanımlanmaktadır:

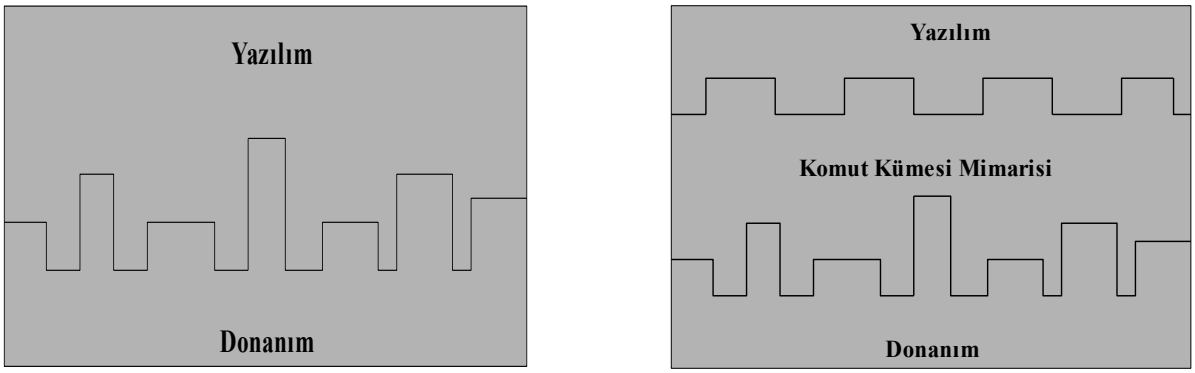
- ISA - Komut kümesi mimarisi
- HSA - Donanım sistem mimarisi

ISA, bir bilgisayarın hesaplama karakteristiklerini belirleyen komut kümesinin tasarımıdır. HSA; CPU, depolama ve G/Ç sistemlerinin dahil olduğu alt sistem ve bunların bağlantı şekilleridir. Komut kümesinin yazılım ve donanımla ilişkisi **Şekil 2.1.'de** görülmektedir.



Şekil 2.1.: Komut kümesinin yazılım ve donanımla ilişkisi

Bilgisayar sistemlerinde bütün mesele,bu iki kavramı yerli yerine oturtmaktır. Mimari bir kavram olarak HSA'nın ne olduğu ve hangi elemanlardan meydana geldiği yukarıda açıklanmıştır. ISA ise, programcının bu elemanlara yön verecek programı yazması durumunda nasıl bir kabul göreceğidir.Farklı şirketler tarafından üretilen farklı bilgisayarların fiyat/performans açısından elbette farklı mimarileri olabilir. Özel bilgisayar sistemleri(günümüzde bir çeşit oyun konsolları)için programcı kodlarını makinanın doğrudan özel donanımına göre yazmaktaydı.Böylece bir makine için yazılan program aynı firma tarafından üretilse bile,ne rekabet ettiği bir makinasında ne de diğer makinasında çalışabilmekteydi. Mesela, A makinası için yazılan bir oyun B makinasında veya C makinasında çalışmayacaktır. Programcı tarafından yazılan kodlar donanımı açma anahtarı olarak düşünülebilir (Şekil 2.2..).



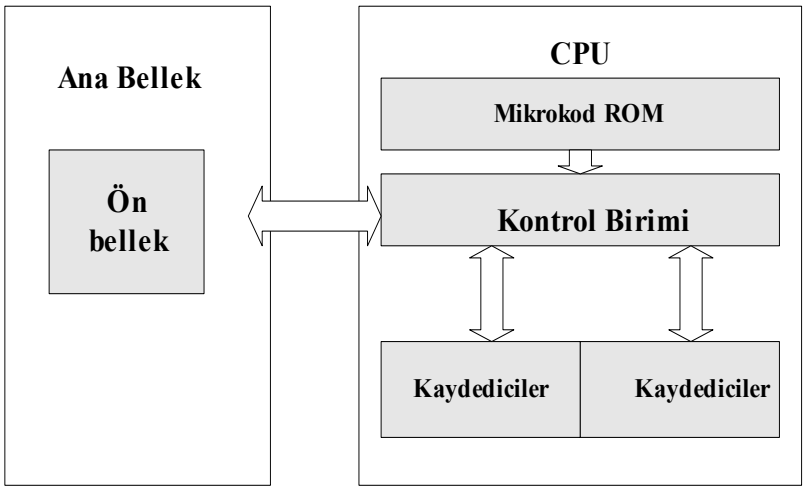
Şekil 2.2.: Komut Kümesi mimarisinin yazılım ve donanımla ilişkisi

Programsal yaklaşım

Bilgisayar sistemlerinde bütün mesele sistemi meydana getiren tüm elemanların bir komutla nasıl devreye sokulacağıdır.Ufak tefek ayrıcalıkları olsa da birbirine benzer yapıdaki bilgisayarlar için farklı programlar yazmak oldukça maliyetli olduğundan ,programcının yazdığı komutların her bilgisayar tarafından algılanarak yürütülmesi esas hedeftir. Ortaya atılan ilk çözüm mikrokod yaklaşımı daha sonraları iki standarttan biri olmuştur.

Donanımı devreye sokacak öz bilgilerin yani komut kümesinin yer aldığı bu yere (bölgeye) **mikrokod motoru** denilmektedir**. (Şekil 2.3.).**

Burası,CPU içinde CPU olarak da ifade edilebilir.Programcının yazdığı kodları işlemcinin daha çabuk anlayabileceği veya çalıştırabileceği küçük mikrokodlara dönüştüren bu mikrokod motoru,işlemci ROM bellek vasıtasıyla yerleştirilmiştir. Mikroprogram ve icra birimi tarafından meydana gelen mikrokod ROM'un görevi,özel komutların bir dizi kontrol sinyallerine çevirerek sistem elemanlarının denetlenmesini sağlar. Aynı zamanda, mikrokod CISC tipi işlemcilerdeki temel işlevi, alt düzey komut kümesiyle programcının çalıştığı üst düzey komutlar arasında soyutlama düzeyi oluşturmaktır.



Şekil 2.3.: Bir mikrokod ROM'un sistemdeki yeri

Mikroişlemci üreticileri,sistem tasarımında iki yönlü düşünmek zorundadırlar. Birincisi,mimariyi meydana getiren elemanların işlevleri,ikincisi bu elemanların nasıl devreye sokulacağıdır. Elemanları devreye sokmak için program yazmak gerekecektir.Bu işin bir yanı; diğer yanı ise donanımdır.Donanımla tasarım mühendisleri ilgilenir.Fakat programcı öyle bir program yazmalı ki,sistem tarafından algılanarak doğru zamanda doğru eleman devreye sokulabilsin. Donanım mimarisini programcıya aktaracak en iyi yol ona kullanabileceği komut kümesini hazır vermektir.Bilgisayar sisteminin donanımsal tüm özelliklerini içeren sisteme **komut kümesi mimarisi** denildiğine göre, programcı bu kümeye bakarak veya bu kümeyi kullanabilen derleyicileri kullanarak hiçbir endişeye gerek duymaz. Programcının yazdığı bir komut işletildiğinde, mikrokod ROM bu komutu okur ve sonra o komuta karşılık gelen uygun mikrokodları yükler ve çalıştır.

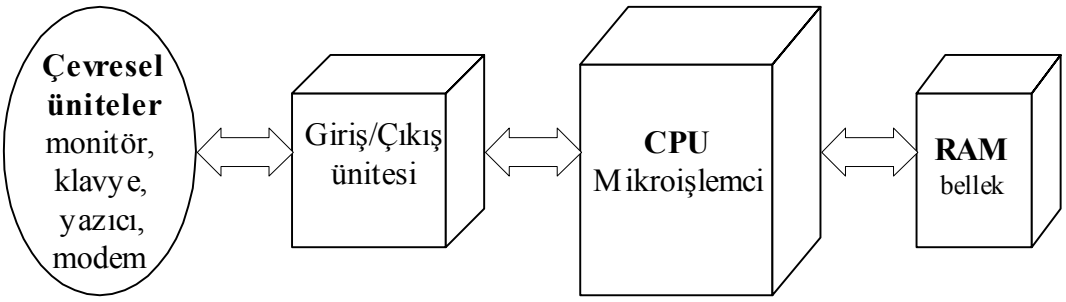
Donanımsal Yaklaşım

Mikrokod kullanılarak ISA sisteminin yürütülmesinin başlıca sakıncası başlangıçta komutların doğrudan çalıştıran sisteme göre yavaş olmasıdır. Mikrokod, ISA tasarımcılarına programcının ara sıra kullandığı her çeşit komutların komut kümesine eklenmesini ister. Daha çok komut demek daha fazla mikrokod, çekirdek büyüklüğü ve güç demektir. ISA mimarisinin yaşanan aksaklıklarından dolayı daha sonraları ,komutların doğrudan donanım elemanları tarafından yorumlanarak sistemin denetlendiği diğer bir mimari yaklaşımda donanımsal çalışma modelidir..Komutların anlaşılır standart bir boyuta getirilerek çalışıldığı sisteme RISC modeli denilmektedir.Yani komutların donanımsal çalışma modeline sahip RISC tipi bilgisayarlarda, komut kümesindeki komutların sayısı azaltılmış ve her bir özel komutun boyutu düşürülmüştür.

2.2. MİKROİŞLEMCİLER VE MİKRODENETLEYİCİLER

MİKROİŞLEMCİ, saklı bir komut dizisini ardışıl olarak yerine getirerek veri kabul edebilen ve bunları işleyebilen sayısal bir elektronik eleman olarak tanımlanabilir. Günümüzde basit oyuncaklardan, en karmaşık kontrol ve haberleşme sistemlerine kadar hemen her şey mikroişlemcili sistemlerle kontrol edilmektedir.

Mikrodenetleyici veya sayısal bilgisayar üç temel kısım (CPU, Giriş/Çıkış Birimi ve Hafıza) ile bunlara ek olarak bazı destek devrelerinden oluşur. Bu devreler en basitten, en karmaşığa kadar çeşitlilik gösterir.



Şekil 2.4 : Bir mikroişlemci sisteminin temel bileşenleri

*Giriş / Çıkış (**I**nput / **O**utput) :* Sayısal, analog ve özel fonksiyonlardan oluşur ve mikroişlemcinin dış dünya ile haberleşmesini sağlar.

*CPU (**C**entral **P**rocessing **U**nit – Merkezi İşlem Birimi) :* Sistemin en temel işlevi ve organizatörüdür. Bilgisayarın beyni olarak adlandırılır.Komutları yürütmek, hesapları yapmak ve verileri koordine etmek için 4, 8, 16, 32 ve 64 bitlik sözcük uzunluklarında çalışır.

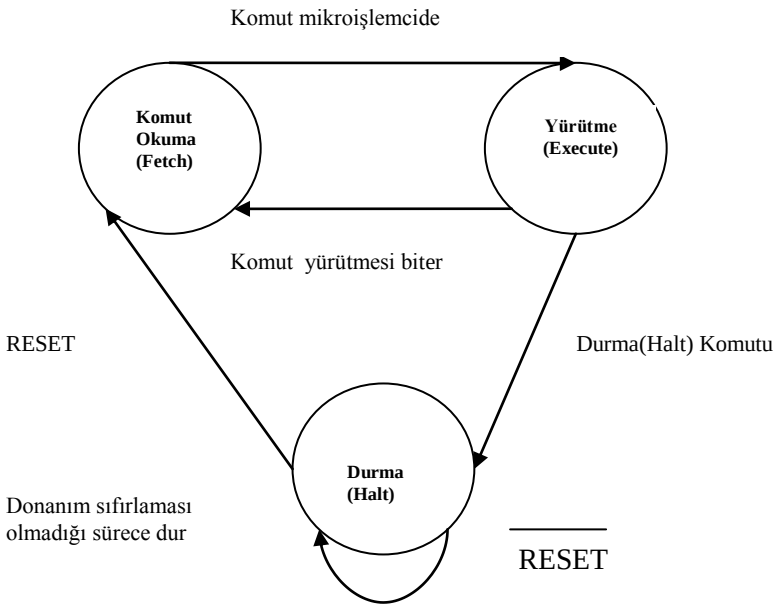
Hafıza : RAM, ROM, PROM, EPROM, EEPROM veya bunların herhangi bir birleşimi olabilir. Bu birim, program ve veri depolamak için kullanılır.

Osilatör : Mikroişlemcinin düzgün çalışabilmesi için gerekli olan elemanlardan biridir. Görevi; veri ve komutların CPU 'ya alınmasında, yürütülmesinde, kayıt edilmesinde, sonuçların hesaplanmasında ve çıktıların ilgili birimlere gönderilmesinde gerekli olan saat darbelerini üretmektir. Osilatör, farklı bileşenlerden oluşabileceği gibi hazır yapılmış bir modül de olabilir.

Diğer devreler : Mikroişlemci ile bağlantılı diğer devreler; sistemin kilitlenmesini önlemeye katkıda bulunan *Watchdog Timer,* mantık aşamalarını bozmadan birden fazla yonganın bir birine bağlanmasını sağlayan adres ve veri yolları (BUS) için *tampon (buffer),* aynı BUS 'a bağlanmış devrelerden birini seçmeyi sağlayan, adres ve I/O için kod çözücü elemanlar *(decoder).*

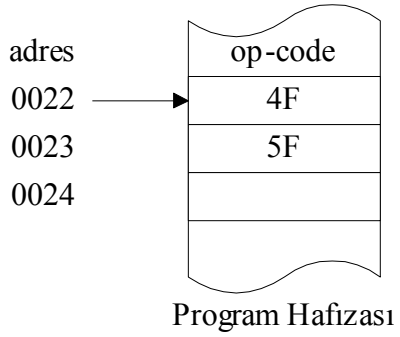
Mikroişlemcinin Çalışması: Bir mikroişlemcinin çalışmasında ,kontrol birimi tarafından yerine getirilen ve **Şekil 2.5'de** gösterilen temel iki işlem vardır. **Komut okuma (fetch)** ve **komut yürütme (execute).** Komut okuma,mikroişlemcinin hafızadan bir **işlem kodu (operation code-opcode)** alıp **komut saklayıcısına (Instruction Register –IR)** getirme işlemine denir. Komut saklayıcısına gelen komut ile hangi işlemin yapılacağı komut kod çözücüsü tarafından belirlenir.Gereken sinyalleme kontrol birimi tarafından üretilir.Eğer komut ile belirlenen işlem için,bazı **işlem verisine (operand)** gerek var ise,bu veriler hafızadan okunur.

Son olarak komutun yürütülmesi gerçekleştirilir. Bir komutun yürütülmesi bittikten sonra, benzeri şekilde; tekrar komut okuma ve yürütme işlemleri,sonsuz bir çevrim içinde, bir **durma (halt)** komutu yürütülünceye kadar yapılır. Mikroişlemcinin çalışmasının durduran bu komut ile işlemci bir üçüncü duruma girer ve bu durumdan çıkabilmesi için bir donanım sıfırlaması **(reset)** gerekir.



Şekil 2.5: Bir mikroişlemcideki komut okuma ve yürütme çevrimleri

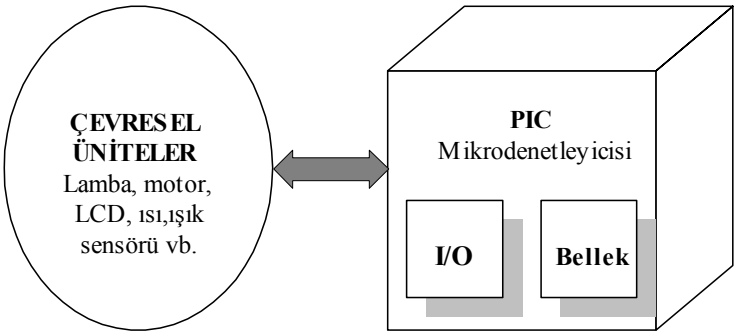
Örnek: Bir mikroişlemcinin program hafızasında bulunan programın çalıştırılması



Şekil-2.6. Program yüklü bulunan hafıza

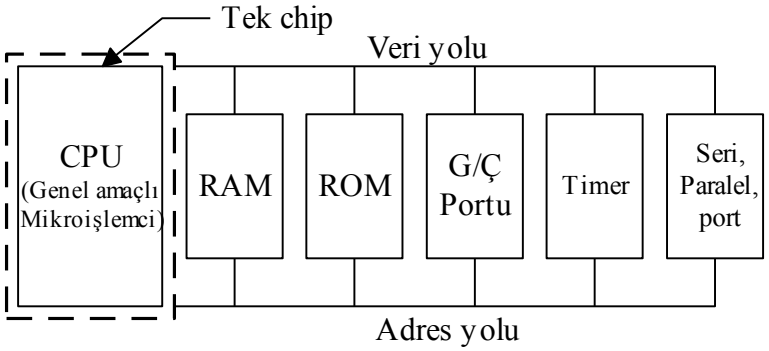
1. Program sayacı o anda çalışan komutun adresini üzerine alır. PC =[0022]
2. Komut kayıtçısı, o anda çalışan komutu üzerinde bulundurur. Ir = 4F
3. Kontrol ünitesi bu komuta göre uygun elektronik sinyalleri uygun yere göndererek komutu çalıştırır. 4F için A akümülatörünü sıfırlayıcı işaret gönderir.
4. Daha sonra PC bir sonraki adrese geçer ve aynı işlemler tekrarlanır.

MİKROBİLGİSAYAR (MİKRODENETLEYİCİ, MİCROCONTROLLER) bir bilgisayar içerisinde bulunması gereken temel bileşenlerden Hafıza, I/O ünitesinin tek bir chip(yonga) üzerinde üretilmiş biçimine denir.

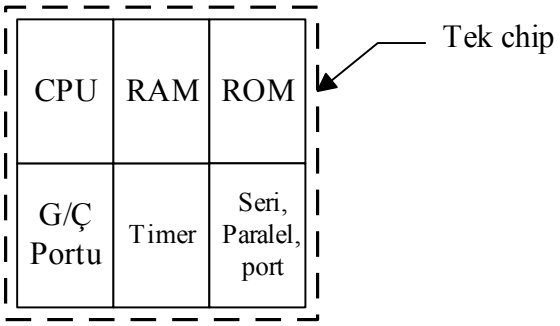


Şekil 2.7 : Bir mikrodenetleyici sisteminin temel bileşenleri

Mikroişlemci ile kontrol edilebilecek bir sistemi kurmak için en azından şu üniteler bulunmalıdır; CPU, RAM, Giriş/çıkış ünitesi ve bu üniteler arasında veri/adres alış verişini sağlamak için bilgi iletim yolları (DATA BUS) gerekmektedir. Bu üniteleri yerleştirmek için baskı devre organizasyonu da önemli bir aşamadır. Mikrodenetleyici ile kontrol edilebilecek sistemde ise yukarıda saydığımız ünitelerin yerine tek bir yonga (mikrodenetleyici) kullanmak yeterli olacaktır. Tek bir yonga kullanmak ile, maliyet düşecek, kullanım ve programlama kolaylığı sağlanacaktır. Bu avantajlardan dolayı son zamanlarda bilgisayar kontrolü gerektiren elektronik uygulamalarda gelişmiş mikroişlemci *(Embeded processor)* kullanma eğilimi gözlenmiştir.



a) Genel amaçlı mikroişlemci sistemi



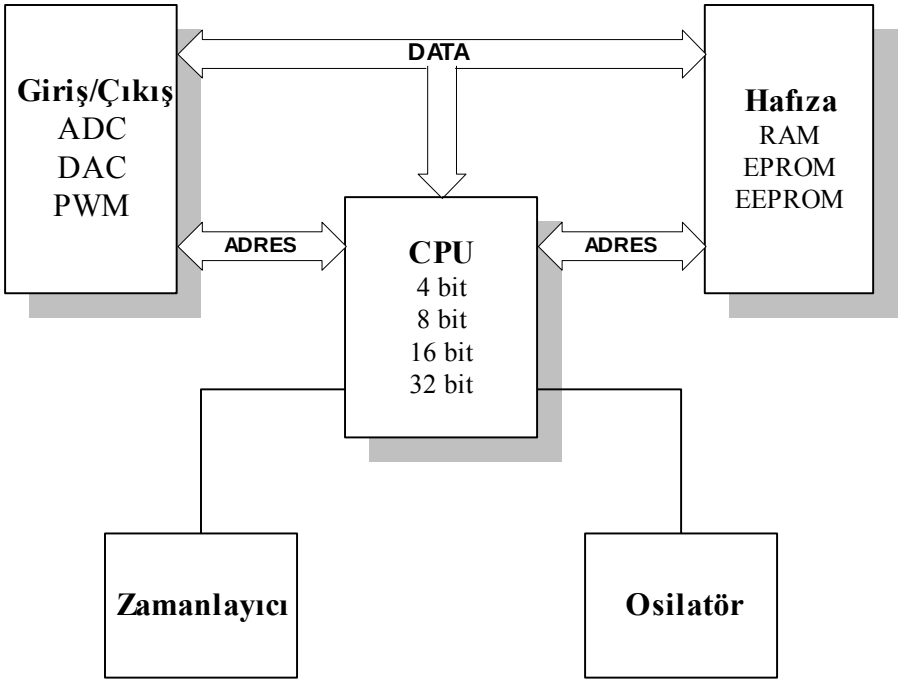
b) Mikrodenetleyici sistemi

Şekil 2.8. Mikroişlemcili sistem ile mikrodenetleyici sistemler

Günümüz mikrodenetleyicileri otomobillerde, kameralarda, cep telefonlarında, fax- modem cihazlarında, fotokopi, radyo, TV, bazı oyuncaklar gibi sayılamayacak kadar pek çok alanda kullanılmaktadır.

Mikrodenetleyiciler 1990'lı yıllardan sonra aşağıdaki ihtiyaçlara cevap verebilmek için gelişmeye başlamışlardır. Gelişim sebepleri;

- Karmaşık cihazlar da daha yüksek performansa ihtiyaç duyulması
- Daha geniş adres alanına sahip olması
- C gibi yüksek seviyedeki dillerde programlama desteğinin sağlanması
- Windows altında çalışan gelişmiş özelliklere sahip program geliştirme ortamlarının olması
- Daha az güç tüketimi ve gürültünün olması
- Büyük geliştirme yatırımları ve yazılım güvenliği açısından varolan çeşitli programların kullanılması
- Sistem fiyatlarının ucuz olması

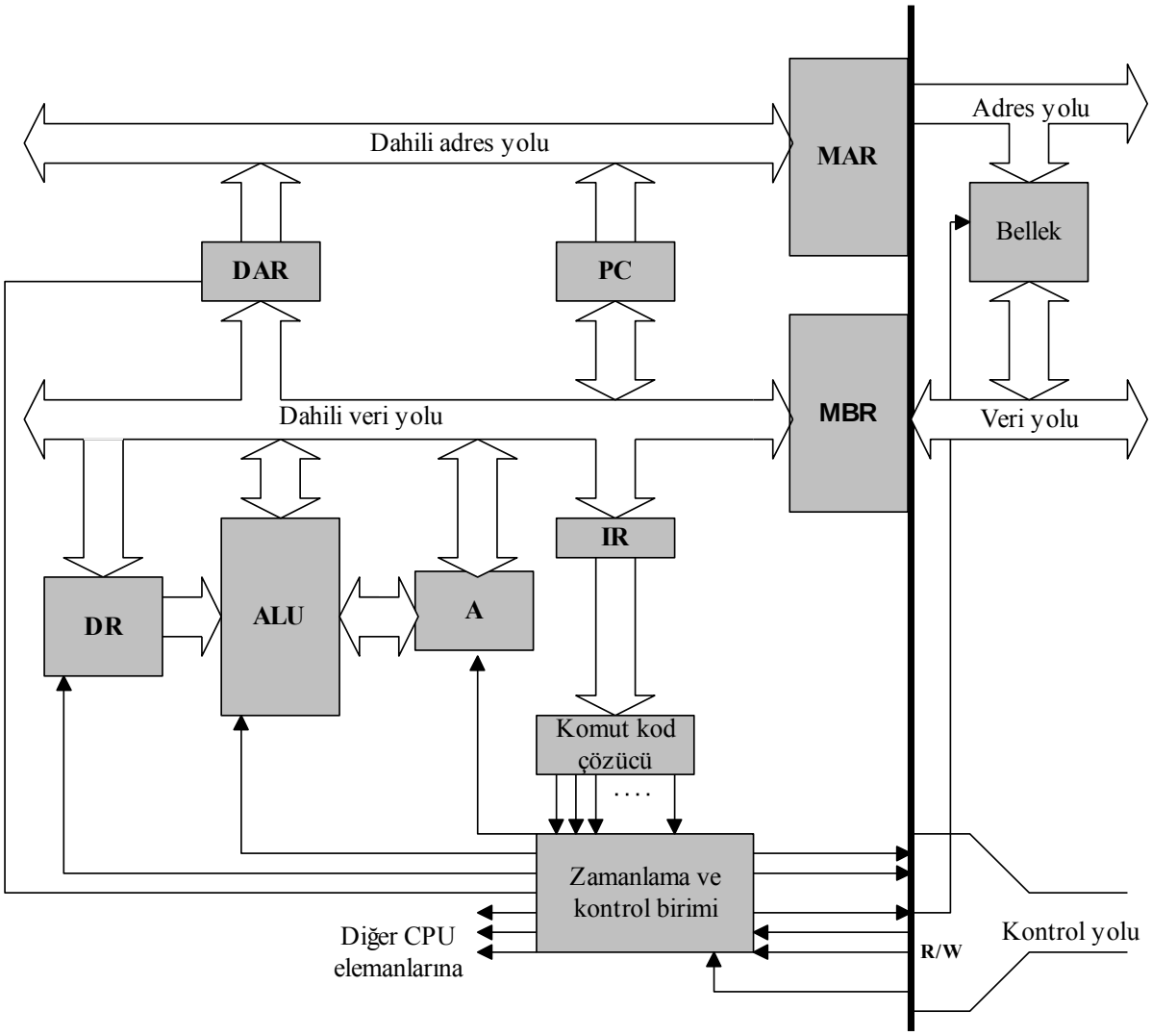


Şekil 2.9: Mikrodenetleyici sistem

Teknolojik gelişmelerle birlikte mikroişlemcilerde zamanla gelişmeye başlamışlardır. Belirli bir sürede ele alınan bit sayısına bakılarak mikroişlemcinin güçlü olup olmadığı belirlenir. Bit uzunluklarına göre 8 bit, 16 bit, 32 bit ve 64 bitlik mikroişlemciler bulunur.

2.3. Basitten Karmaşığa Mikroişlemci Yapısı

2.3.1. 8-Bitlik Mikroişlemciler: Basit bir işlemci kaydediciler, aritmetik-mantık birimi ve denetim birimi olmak üzere 3 ana bölümden meydana gelmiştir.

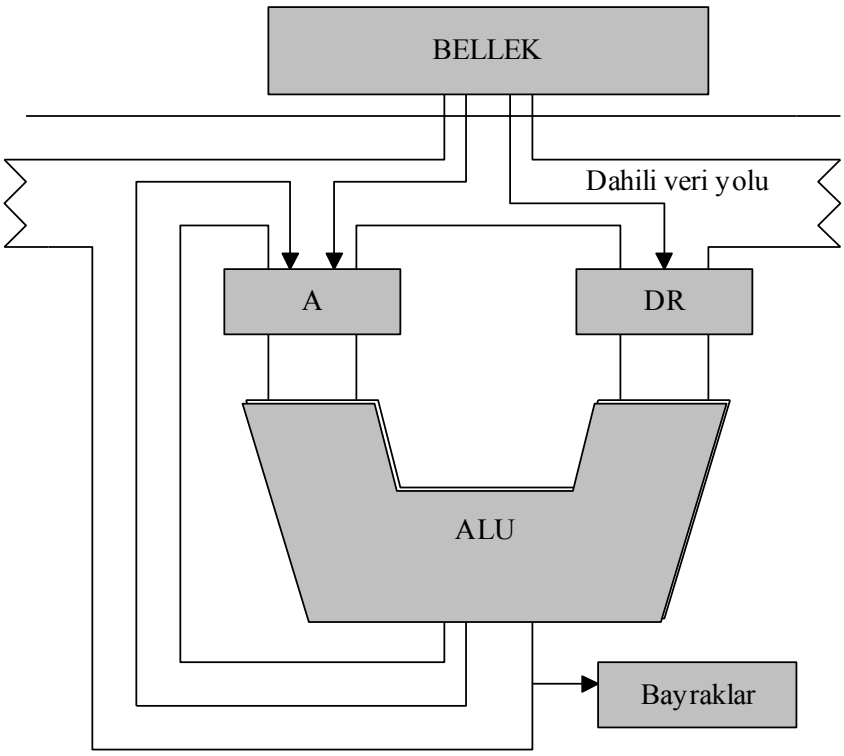


Şekil 2.10 : Basit bir 8-bitlik işlemcinin yapısını oluşturan ana birimler

Kaydediciler: Flip-floplardan oluşan birimlerdir. İşlemci içerisinde olduklarından belleklere göre daha hızlı çalışırlar. İşlemci çeşitlerine göre kaydedicilerin adı ve tipleri değişmektedir. Kaydediciler genel amaçlı ve özel amaçlı olmak üzere iki grupta incelenmektedir.Genel amaçlı kaydediciler grubuna A, B ve X gibi kaydediciler girer. A kaydedicisi Akümülatör teriminden dolayı bu adı almıştır. İndis kaydedicilerinin görevleri ise; hesaplamalar sırasındaki ara değerlerin üzerinde tutulması, döngülerde sayaç olarak kullanılmasıdır. Özel amaçlı kaydediciler ise; PC (Program Counter, Program Sayacı), SP (Stack Pointer- Yığın İşaretçisi) ve Flags (Bayraklar) verilebilir. Bunların dışında işlemcide programcıya görünmeyen kaydediciler vardır. Bu kaydedicileri alt düzey program yazan programcıların mutlaka bilmesi gerekir. Bunlar; IR(Instruction Register-Komut kaydedicisi), MAR (Memory Address Register- Bellek adres

kaydedicisi), MBR (Memory Buffer Register- Bellek veri kaydedicisi), DAR(Data Address Register- Veri adres kaydedicisi) ve DR (Data register- Veri kaydedicisi) olarak ele alınabilir.

Aritmetik ve Mantık Birimi:ALU mikroişlemcilerde aritmetiksel ve mantıksal işlemlerinin yapıldığı en önemli birimdir. Aritmetiksel işlemler denilince akla başta toplama, çıkarma, çarpma ve bölme gelir. Komutlarla birlikte bu işlemleri, mantık kapıları, bu kapıların oluşturduğu toplayıcılar, çıkarıcılar ve flipfloplar gerçekleştirir. Mantıksal işlemlere de AND, OR, EXOR ve NOT gibi işlemleri örnek verebiliriz.

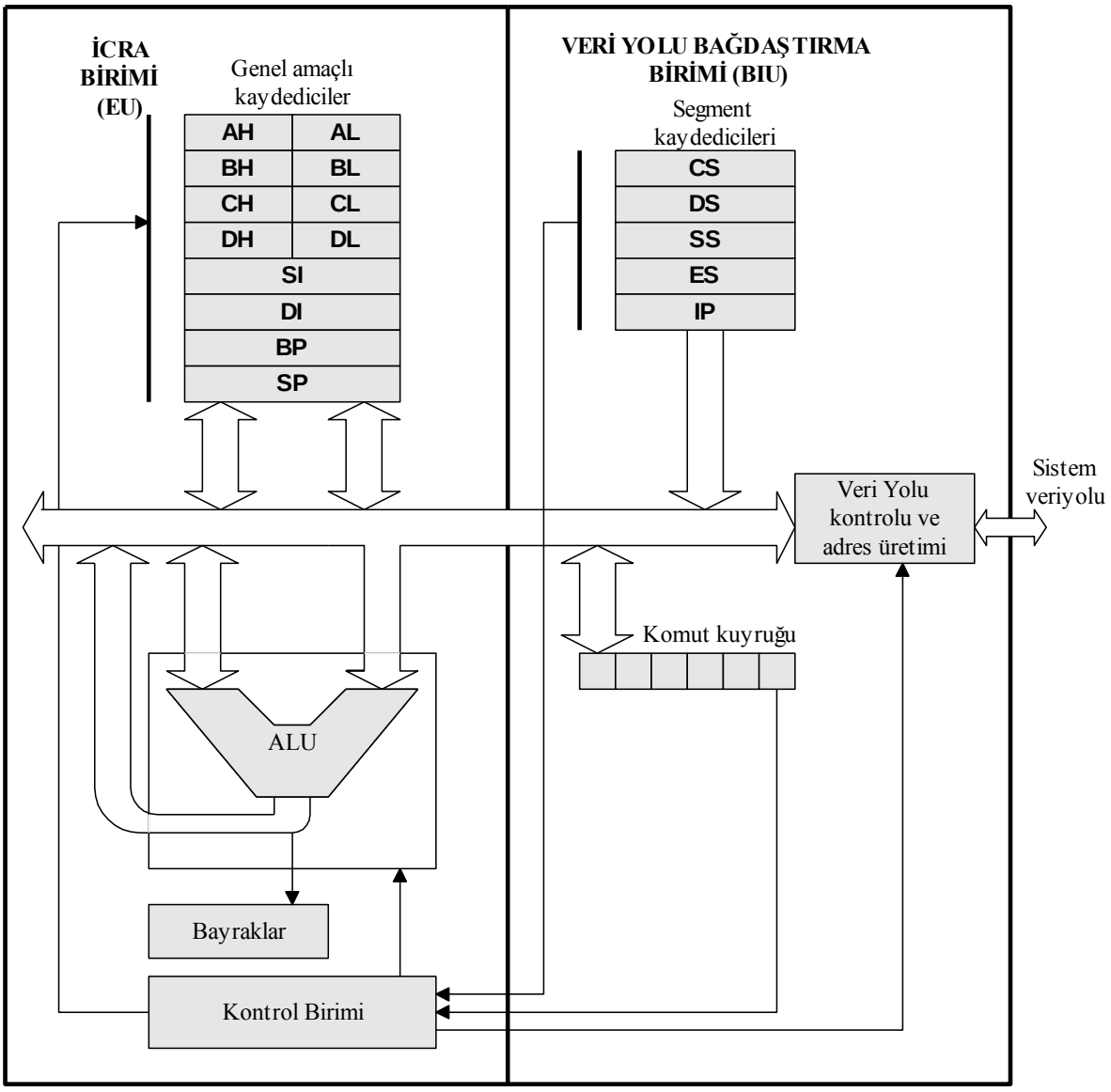


Şekil 2.11 : Aritmetik ve mantık birimi

Zamanlama ve Denetim Birimi: Bu kısım sitemin tüm işleyişinden ve işlemin zamanında yapılmasından sorumlu olan birimdir. Bu birim bellekte program bölümünde bulunan komut kodunun alınıp getirilmesi, kodunun çözülmesi, ALU tarafından işlenip, sonucun alınıp belleğe yüklenmesi için gerekli olan denetim sinyalleri üretir.

İletişim yolları: Mikroişlemci mimarisine girmese de işlemciyle ayrılmaz bir parça oluşturan iletişim yolları kendi aralarında üçe ayrılır. Adres yolu; komut veya verinin bellekte bulunduğu adresten alınıp getirilmesi veya adres bilgisinin saklandığı yoldur. Veri yolu ise işlemciden belleğe veya Giriş/Çıkış birimlerine veri yollamada yada tersi işlemlerde kullanılır. Kontrol yolu ise sisteme bağlı birimlerin denetlenmesini sağlayan özel sinyallerin oluşturduğu bir yapıya sahiptir.

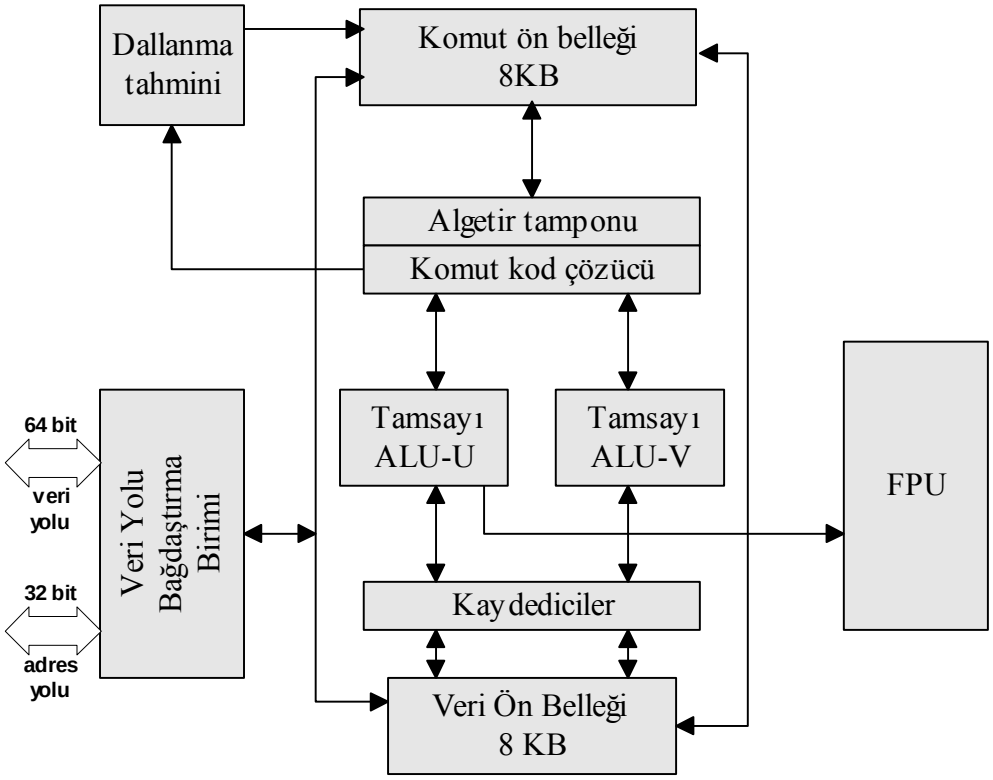
2.3.2. 16-Bitlik Mikroişlemciler: 16-bitlik mikroişlemciler basit olarak 8- bitlik mikroişlemcilerde olduğu gibi , Kaydediciler, ALU ve Zamanlama-Kontrol birimine sahiptir. Fakat mimari yapısı çoklu görev ortamına uygun hale getirildiğinden, işlemci içerisindeki bölümlerde fonksiyonel açıdan 2 mantıksal bölümden oluşurlar. Bu birimler Veri Yolu Bağdaştırma Birimi (BIU) ve İcra Birimi (EU) 'dir. BIU birimi, EU birimini veriyle beslemekten sorumluyken, icra birimi komut kodlarının çalıştırılmasından sorumludur. BIU bölümüne segment kaydedicileriyle birlikte IP ve komut kuyrukları ve veri alıp getirme birimleri dahilken, EU bölümüne genel amaçlı kaydediciler, kontrol birimi, aritmetik ve mantıksal komutların işlendiği birim dahildir.



Şekil 2.12: 16- bitlik mikroişlemci mimarisi

2.3.3. 32-Bitlik Mikroişlemciler: 3. kuşak mikroişlemcilerdir. Diğerlerinden farklı olarak içerisine FPU (Floating Point Unit- Kayan nokta birimi) denilen ve matematik işlemlerinden sorumlu olan bir birim eklenmiştir. Bu gelişmiş işlemci 64-bitlik geniş bir harici veri yoluna sahiptir. Geniş veri yolu, işlemcinin bir çevrimlik zamanda daha çok veri taşıması ve dolayısıyla yapacağı görevi daha

kısa zamanda yapması demektir. Bu, işlemcinin bir tıklanmasıyla, işlemci ile bellek arasında veya işlemci ile G/Ç birimleri arasında, 8-bitlik bir işlemciye göre 8 kat fazla bilgi taşınması demektir.



Şekil 2.13 : 32-bitlik mikroişlemci mimarisi

2.4. Mikrodenetleyicilerin (Mikrobilgisayarların) Gelişimi

CPU, Bellek ve Giriş/Çıkış birimlerinin bir arada bulunması mikrodenetleyiciyi özellikle endüstriyel kontrol uygulamalarında güçlü bir dijital işlemci haline getirmiştir. Mikrodenetleyiciler özellikle otomobillerde motor kontrol, elektrik ve iç panel kontrol; kameralarda, ışık ve odaklama kontrol gibi amaçlarda kullanılmaktadır. Bilgisayarlar, telefon ve modem gibi çeşitli haberleşme cihazları, CD teknolojisi, fotokopi ve faks cihazları, radyo, TV, teyp, oyuncaklar, özel amaçlı elektronik kartlar ve sayılmayacak kadar çok alanda , mikrodenetleyiciler kullanılmaktadır. Bu kadar geniş bir uygulama alanı olan mikrodenetleyiciler aşağıda sıralanan çeşitli özelliklere sahiptirler.

- Programlanabilir sayısal paralel giriş / çıkış
- Programlanabilir analog giriş / çıkış
- Seri giriş / çıkış (senkron, asenkron ve cihaz denetimi gibi)
- Motor/servo kontrolü için darbe işaret çıkışı (PWM gibi)
- Harici giriş ile kesme
- Zamanlayıcı (Timer) ile kesme
- Harici bellek arabirimi
- Harici BUS arabirimi (PC ISA gibi)
- Dahili bellek tipi seçenekleri (ROM, PROM, EPROM ve EEPROM)

- Dahili RAM seçeneği
- Kesirli sayı (kayan nokta) hesaplaması
- D/A ve A/D çeviricileri

Bu özellikler mikrodenetleyicileri üreten firmalara ve mikrodenetleyicilerin tipine göre değişmektedir.

Mikrodenetleyici uygulamalarında dikkate alınması gereken en önemli özellikler *gerçek zaman (*real time) işlemi ve çok görevlilik (multi-tasking) özellikleridir. Gerçek zaman işlemi, mikrodenetleyicinin ihtiyaç anında çalışma ortamına, gereken kontrol sinyallerini göndermesi ve ortamı bekletmeyecek kadar hızlı olmasıdır. Çok görevlilik ise mikrodenetleyicinin birçok görevi aynı anda veya aynı anda gibi yapabilme kapasitedir. Mikrodenetleyici özet olarak kullanıldığı sistemin birçok özelliğini aynı anda *gözleme (monitoring)*, ihtiyaç anında *gerçek-zamanda cevap verme (real-time response*) ve sistemi *denetlemeden (control)* sorumludur.

Bir çok firma tarafından mikrodenetleyiciler üretilmektedir. Her firma üretmiş olduğu mikrodenetleyici yongaya farklı isimler ve özelliklerini birbirinden ayırmak içinde parça numarası vermektedir. Bu denetleyicilerin mimarileri arasında çok küçük farklar olmasına rağmen aşağı yukarı aynı işlemleri yapabilmektedir. Her firma ürettiği chip'e bir isim ve özelliklerini biri birinden ayırmak içinde parça numarası vermektedir. Günümüzde yaygın olarak 8051(intel firması) ve PIC adı verilen mikrodenetleyiciler kullanılmaktadır. Bunlardan başka Phillips, Dallas, Siemens, Oki, Temic, Haris, Texas gibi çeşitli firmalarda üretim yapmaktadır. Örneğin; bunlardan *Microchip* firması üretmiş olduklarına *PIC* adını verirken, parça numarası olarak da *12C508, 16C84, 16F877* gibi kodlamalar vermiştir, *Intel* ise ürettiği mikrodenetleyicilere *MCS-51* ailesi adını vermiştir, *Texas Ins.* ise işaret işlemeye yönelik olarak *Digital Signal Processing (DSP)* mikrodenetleyici yongası üretmektedir. PIC mikrodenetleyicileri elektronik cihazlarda çok yaygın olarak kullanılmaktadır. Çünkü her amaca uygun boyut ve özellikte mikrodenetleyici bulmak mümkündür. Çeşitli tiplerde mikrodenetleyiciler kullanılabilir fakat, uygulamada en küçük, en ucuz, en çok bulunan ve yapılan işin amacına yönelik olmasına dikkat edilmelidir. Bunun içinde mikrodenetleyicilerin genel özelliklerinin iyi bilinmesi gerekir.

Mikrodenetleyicili bir sistemin gerçekleştirile bilinmesi için, mikrodenetleyicinin iç yapısının bilinmesi kadar, sistemin yapacağı iş için mikrodenetleyicinin programlanması da büyük bir önem arz eder. Mikrodenetleyicili sistemler ile bir oda sıcaklığını, bir motorun hızını, led ışık gibi birimlerini kontrol edebiliriz. Bütün bu işlemleri nasıl yapacağını mikrodenetleyiciye tarif etmek, açıklamak gerekir. Bu işlemlerde mikrodenetleyicili sistemin program belleğine yerleştirilen programlar vasıtasıyla gerçekleştirilir. Mikrodenetleyiciler için programlar asembly veya C gibi bir programlama dilinde yazılabilir. Assembly dilinde yazılan bir program assembler adı verilen bir derleyici ile makine diline çevrildikten sonra mikrodenetleyiciye yüklenir. C dilinde yazılan programında bir çevirici ile makine diline çevrilmesi gerekmektedir. Makine dilindeki bir

programın uzantısı '.HEX' dir. PIC mikrodenetleyicisi için program yazarken editör ismi verilen bir programa ihtiyaç vardır. En çok kullanılan editör programı ise MPLAB'tır. MPLAB' da yazılan programlar proje dosyalarına dönüştürülerek, aynı editör içerisinde derlenebilmektedir.

Bütün bu özellikler dikkate alınarak en uygun mikrodenetleyici seçimi yapılmalıdır. Çünkü mikrodenetleyiciler ticari amaçlı birçok elektronik devrede yaygın olarak kullanılmaktadır.

2.5. Mikrodenetleyici Seçimi

Mikrodenetleyici seçimi kullanıcı için oldukça önemlidir, çünkü mikrodenetleyiciler ticari amaçlı bir elektronik devrede yaygın olarak kullanılmaktadır. Bu sistemlerin öncelikle maliyetinin düşük olması için mikrodenetleyicinin de ufak ve ucuz olması istenir. Diğer taraftan ürünün piyasada bol miktarda bulunması da önemlidir. Tüm bu hususlar dikkate alınarak, kullanıcılar öncelikle hangi firmanın ürününü kullanacağına karar verirler. Daha sonra da hangi seriden, hangi ürünün kullanacaklarına karar verirler. Burada mikrodenetleyicinin belleğinin yazılım için yeterli büyüklükte olması, kullanılması düşünülen ADC (Analog Dijital Dönüştürücü) kanalı, port sayısı, zamanlayıcı sayısı ve PWM (Pulse Widht Modulation- Darbe Genişlik Modülasyonu) kanalı sayısı önemlidir. Ayrıca tasarımcı yapılacak iş için uygun hızda mikrodenetleyici kullanmalıdır. Tüm bu hususlar dikate alınarak uygun mikrodenetleyiciye karar verilir. Ürün geliştirmek için pencereli (EPROM) veya FLASH tipinde olan belleği silinip, yazılabilen mikrodenetleyici kullanılır. Çünkü ürün geliştirme aşamasında mikrodenetleyici defalarca silinip, yazılabilmektedir. Ayrıca belleği daha hızlı silinip, yazılabilen FLASH mikrodenetleyiciler öğrenmeye yeni başlayanlar için cazip olmaktadır.

Seçimi etkileyen bu noktaları kısaca açıklarsak;

- *Mikrodenetleyicinin İşlem Gücü:* Her uygulamada farklı bir işlem gücüne gereksinim duyulabilir. Bunun için yapılacak uygulamada kullanılacak mikrodenetleyicinin çalışabileceği en yüksek frekans aralığı seçilmelidir.
- *Belleğin Kapasitesi ve Tipi:* Geliştirilecek olan uygulamaya göre program belleği, veri belleği ve geçici bellek kapasitesi dikkate alınmalıdır. Kullanılacak olan belleğin tipide uygulama için önemli bir faktördür.
- *Giriş/Çıkış Uçları:* Mikrodenetleyicinin çevre birimler ile haberleşmesinin sağlayan uçlardır. Bu nedenle giriş/ çıkış uçlarının sayısı oldukça önemlidir. Yapılacak olan uygulamaya göre bu faktörde dikkate alınmalıdır.
- *Özel Donanımlar:* Yapılacak olan uygulamanın çeşidine göre mikrodenetleyiciye farklı çevre birimleri de eklenebilir. Mikrodenetleyici çevre birimleri ile iletişim kurarken kullanacağı seri, I^2C, SPI, USB, CAN gibi veri iletişim protokollerini destekleyen veya ADC, analog karşılaştırıcı gibi analog verileri işleyebilecek donanımlara sahip olması dikkate alınmalıdır.

- *Kod Koruması:* Mikrodenetleyicinin sahip olduğu kod koruması özellikle ticari uygulamalarda program kodunun korunmasına olanak sağlamaktadır.

2.6. Bölüm Kaynakları

1. *O. Altınbaşak, 2001. "Mikrodenetleyiciler ve PIC Programlama", Atlaş Yayıncılık, İstanbul.*
2. *N. Gardner, 1998. "PIC Programlama El Kitabı", Bileşim Yayıncılık, İstanbul.*
3. *O. Urhan, M.Kemal Güllü, 2004. "Her Yönüyle PIC16F628", Birsen Yayınevi, İstanbul.*
4. *N. Topaloğlu, S. Görgünoğlu,2003. "Mikroişlemciler ve Mikrodenetleyiciler", Seçkin Yayıncılık, Ankara.*
5. *Y. Bodur, 2001. "Adım Adım PICmicro Programlama",İnfogate.*
6. *M. Kemal Güngör,2003. "Endüstriyel Uygulamalar İçin Programlanabilir Kontrol Ünitesi" .*

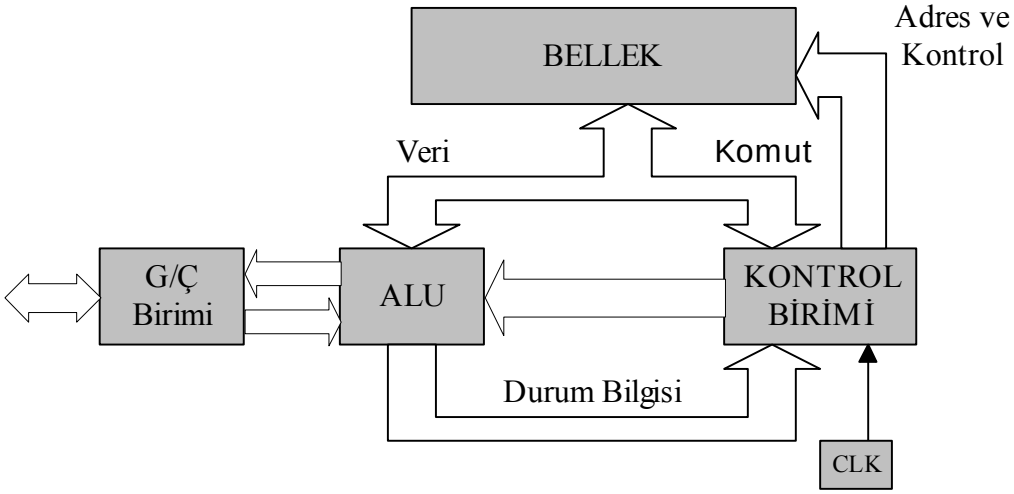
BÖLÜM 3. MİMARİLER

3.1. Mikrodenetleyici / Mikrobilgisayar Tasarım Yapıları

Bilgisayarın yüklenen tüm görevleri çok kısa zamanda yerine getirmesinde yatan ana unsur bilgisayarın tasarım mimarisidir. Bir mikroişlemci, mimari yetenekleri ve tasarım felsefesiyle şekillenir.

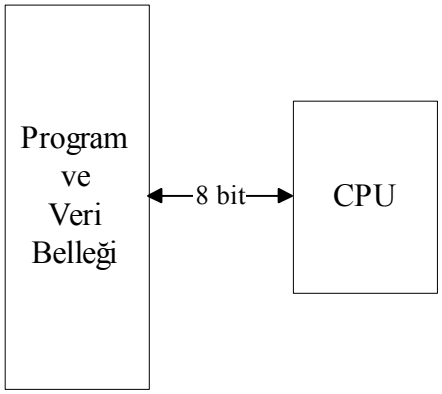
3.1.1. Von Neuman (Princeton) Mimarisi

Bilgisayarlarda ilk kullanılan mimaridir. İlk bilgisayarlar Von Neuman yapısından yola çıkılarak geliştirilmiştir. Geliştirilen bu bilgisayar beş birimden oluşmaktaydı. Bu birimler; aritmetik ve mantıksal birim, kontrol birim, bellek, giriş-çıkış birimi ve bu birimler arasında iletişimi sağlayan yolardan oluşur.



Şekil 3.1. Von Neuman mimarili bilgisayar sistemi

Bu mimaride veri ve komutlar bellekten tek bir yoldan mikroişlemciye getirilerek işlenmektedir. Program ve veri aynı bellekte bulunduğundan, komut ve veri gerekli olduğunda aynı iletişim yolunu kullanmaktadır. Bu durumda, komut için bir algetir saykılı, sonra veri için diğer bir algetir saykılı gerekmektedir.



Şekil 3.2. Von Neuman mimarisi

Von Neuman mimarisine sahip bir bilgisayar aşağıdaki sıralı adımları gerçekleştirir.

1. Program sayıcısının gösterdiği adresten (bellekten) komutu algetir.
2. Program sayıcısının içeriğini bir artır.
3. Getirilen komutun kodunu kontrol birimini kullanarak çöz. Kontrol birimi, bilgisayarın geri kalan birimlerine sinyal göndererek bazı operasyonlar yapmasını sağlar.
4. 1. adıma geri dönülür.

Örnek 3.1:

Mov acc, reg

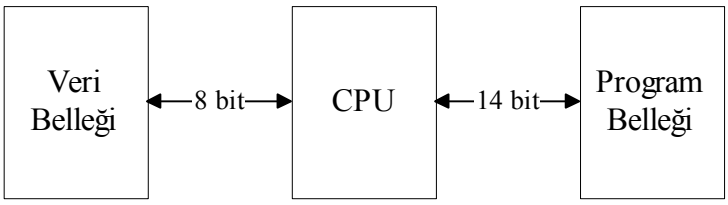
1. cp : Komut okur

2.,.. cp : Veriyi okur ve *acc* ye atar.

Von Neuman mimarisinde, veri bellekten alınıp işledikten sonra tekrar belleğe gönderilmesinde çok zaman harcanır. Bundan başka, veri ve komutlar aynı bellek biriminde depolandığından, yanlışlıkla komut diye veri alanından kod getirilmesi sıkıntılara sebep olmaktadır. Bu mimari yaklaşıma sahip olan bilgisayarlar günümüzde, verilerin işlenmesinde, bilginin derlenmesinde ve sayısal problemlerde olduğu kadar endüstriyel denetimlerde başarılı bir şekilde kullanılmaktadır.

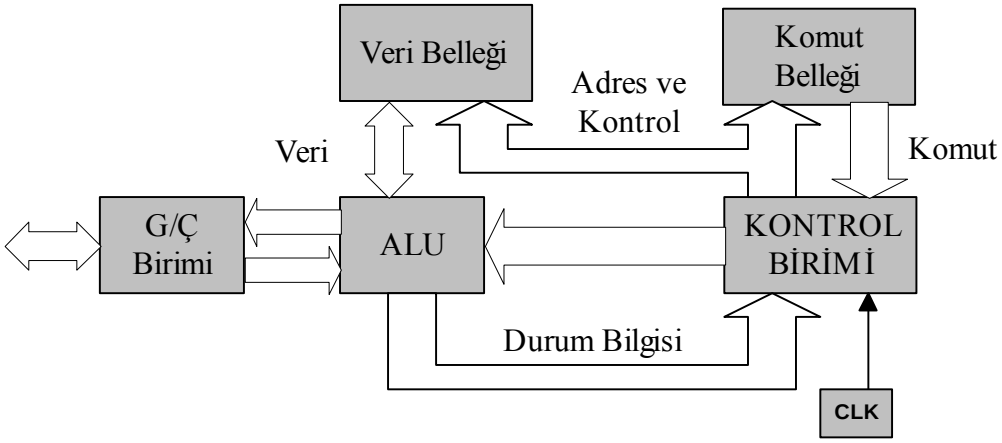
3.1.2. Harvard Mimarisi

Harvard mimarili bilgisayar sistemlerinin Von Neuman mimarisinden farkı veri ve komutların ayrı ayrı belleklerde tutulmasıdır. Buna göre, veri ve komut aktarımında iletişim yolları da bir birinden bağımsız yapıda bulunmaktadırlar.



Şekil 3.3. Harvard Mimarisi

Komutla birlikte veri aynı saykıl da farklı iletişim yolundan ilgili belleklerden alınıp işlemciye getirilebilir. Getirilen komut işlenip ilgili verisi veri belleğinden alınırken sıradaki komut, komut belleğinden alınıp getirilebilir. Bu önden alıp getirme işlemi, dallanma haricinde hızı iki katına çıkarabilmektedir.



Şekil 3.4. Harvard Mimarili bilgisayar sistemi

Örnek 3.2:

Mov acc, reg

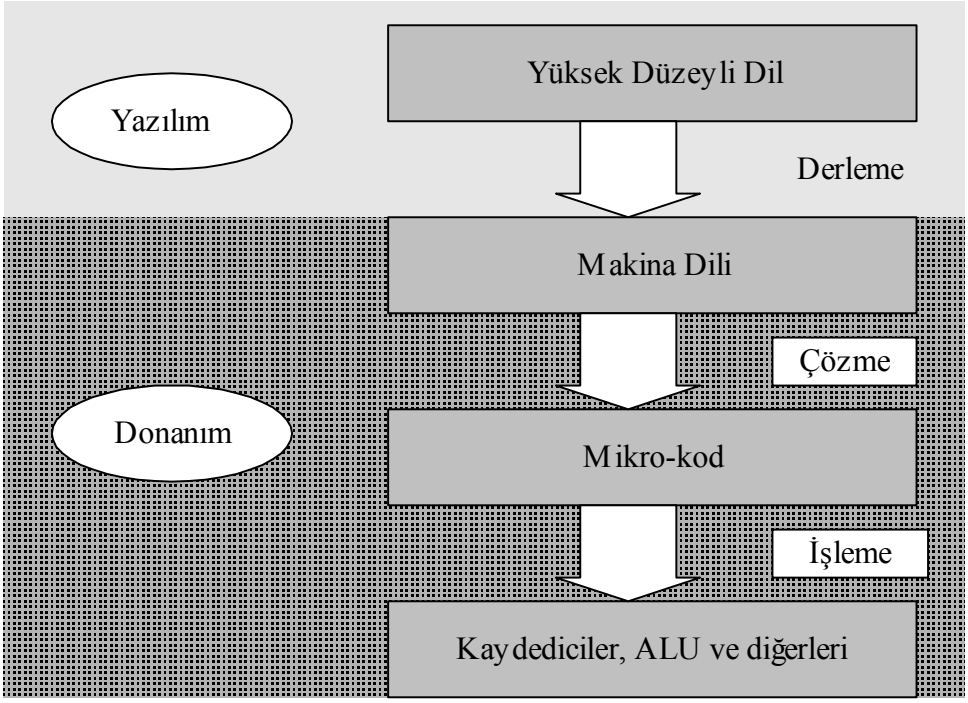
1. cp : Öncelikle *"move acc, reg"* komutunu okur.
2. cp : Sonra *"move acc, reg"* komutunu yürütür.

Bu mimari günümüzde daha çok sayısal sinyal işlemcilerinde (DSP) kullanılmaktadır. Bu mimaride program içerisinde döngüler ve zaman gecikmeleri daha kolay ayarlanır.Von Neuman yapısına göre daha hızlıdır. Özellikle PIC mikrodenetleyicilerinde bu yapı kullanılır.

3.2. Mikroişlemci Komut Tasarım Mimarileri

3.2.1. CISC (Complex Instruction Set Computer) Mimarisi

Bu mimari, programlanması kolay ve etkin bellek kullanımı sağlayan tasarım felsefesinin bir ürünüdür. İşlemci üzerinde performans düşüklüğü ve işlemcinin karmaşık bir hale gelmesine neden olsa da yazılımı basitleştirmektedir. Bu mimarinin en önemli iki özelliği, değişken uzunluktaki komutlar diğeri ise karmaşık komutlardır. Değişken ve karmaşık uzunluktaki komutlar bellek tasarrufu sağlar. Karmaşık komutlar birden fazla komutu tek bir hale getirirler. Karmaşık komutlar aynı zamanda karmaşık bir mimariyi de oluşturur. Mimarideki karışıklık işlemcinin performansını da doğrudan etkilemektedir. Bu sebepten dolayı çeşitli istenmeyen durumlar ortaya çıkabilir. CISC komut seti mümkün olabilen her durum için bir komut içermektedir. CISC mimarisinde yeni geliştirilen bir mikroişlemci eski mikroişlemcilerin assembly dilini desteklemektedir.



Şekil 3.5. CISC tabanlı bir işlemcinin çalışma biçimi

CISC mimarisi çok kademeli işleme modeline dayanmaktadır. İlk kademe, yüksek seviyeli dilin yazıldığı yerdir. Sonraki kademeyi ise makine dili oluşturur. Burada yüksek seviyeli dilin derlenmesi ile bir dizi komutlar makine diline çevrilir. Bir sonraki kademede makine diline çevrilen komutların kodları çözülerek , mikrokodlara çevrilir. En son olarak da işlenen kodlar gerekli olan görev yerlerine gönderilir.

CISC Mimarisinin Avantajları

- Mikroprogramlama assembly dilinin yürütülmesi kadar kolaydır ve sistemdeki kontrol biriminden daha ucuzdur.
- Yeni geliştirilen mikrobilgisayar bir öncekinin assembly dilini desteklemektedir.
- Verilen bir görevi yürütmek için daha az komut kullanılır. Böylece bellek daha etkili kullanılır.
- Mikroprogram komut kümeleri, yüksek seviyeli dillerin yapılarına benzer biçimde yazıldığından derleyici karmaşık olmak zorunda değildir.

CISC Mimarisinin Dezavantajları

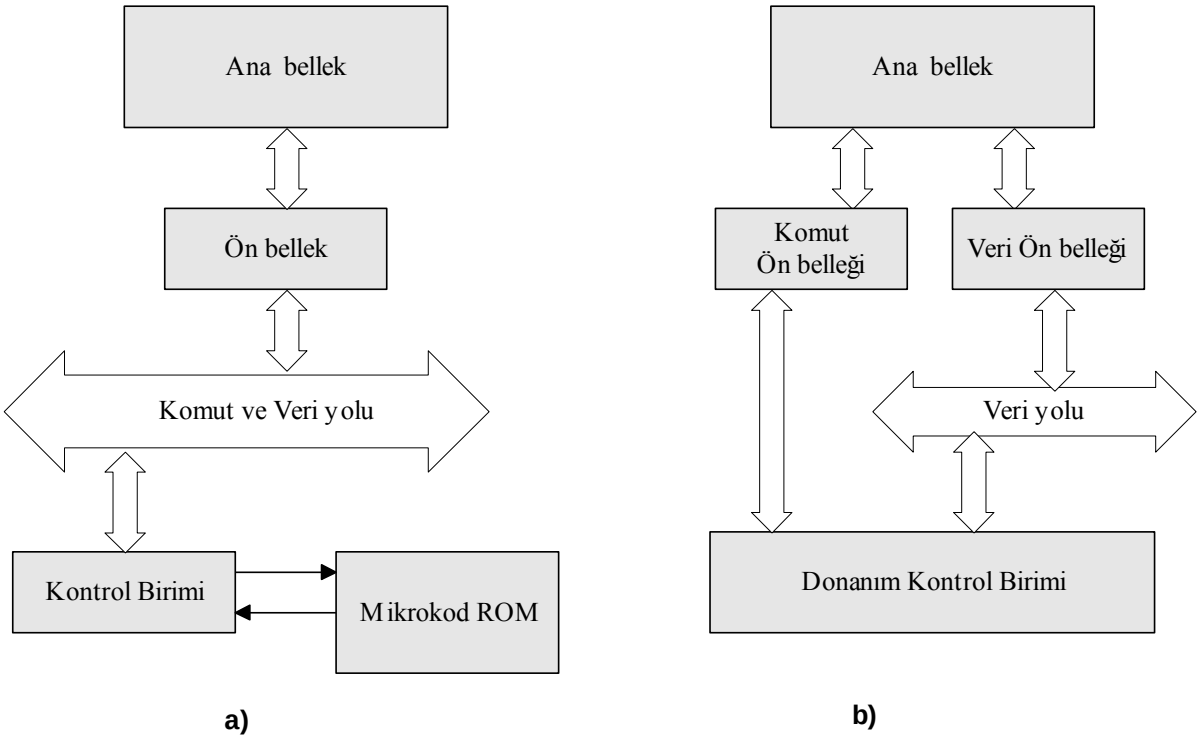
- Gelişen her mikroişlemci ile birlikte komut kodu ve yonga donanımı daha karmaşık bir hale gelmiştir.
- Her komutun çevirim süresi aynı değildir. Farklı komutlar farklı çevrim sürelerinde çalıştıkları için makinanın performansını düşürecektir.
- Bir program içerisinde mevcut komutların hepsi kullanılamaz.

- Komutlar işenirken bayrak bitlerinin dikkat edilmesi gerekir. Buda ek zaman süresi demektir. Mikroişlemcinin çalışmasını etkilemektedir.

3.2.2. RISC (Reduced Instruction Set Computer) Mimarisi

RISC mimarisi IBM, Apple ve Motorola gibi firmalarca sistematik bir şekilde geliştirilmiştir. RISC mimarisinin taraftarları, bilgisayar mimarisinin gittikçe daha karmaşık hale geldiğini ve hepsinin bir kenara bırakılıp en başta yeniden başlamak fikrindeydiler. 70'li yılların başında IBM firması ilk RISC mimarisini tanımlayan şirket oldu. Bu mimaride bellek hızı arttığından ve yüksek seviyeli diller assembly dilinin yerini aldığından, CISC'in başlıca üstünlükleri geçersiz olmaya başladı. RISC'in felsefesi üç temel prensibe dayanır.

- *Bütün komutlar tek bir çevrimde çalıştırılmalıdır:* Her bir komutun farklı çevrimde çalışması işlemci performansını etkileyen en önemli nedenlerden biridir. Komutların tek bir çevrimde performans eşitliğini sağlar.
- *Belleğe sadece "load" ve "store" komutlarıyla erişilmelidir.* Eğer bir komut direkt olarak belleği kendi amacı doğrultusunda yönlendirilirse onu çalıştırmak için birçok saykıl geçer. Komut alınıp getirilir ve bellek gözden geçirilir. RISC işlemcisiyle, belleğe yerleşmiş veri bir kaydediciye yüklenir, kaydedici gözden geçirilir ve son olarak kaydedicinin içeriği ana belleğe yazılır.
- *Bütün icra birimleri mikrokod kullanmadan donanımdan çalıştırılmalıdır.* Mikrokod kullanımı, dizi ve benzeri verileri yüklemek için çok sayıda çevrim demektir. Bu yüzden tek çevirimli icra birimlerinin yürütülmesinde kolay kullanılmaz.



Şekil 3.6. a) Mikrokod denetimli CISC mimarisi; **b)** Donanım denetimli RISC mimarisi

RISC mimarisi küçültülen komut kümesi ve azaltılan adresleme modları sayısı yanında aşağıdaki özelliklere sahiptir.

- Bir çevrimlik zamanda komut işleyebilme
- Aynı uzunluk ve sabit formatta komut kümesine sahip olma
- Ana belleğe sadece "load" ve "store" komutlarıyla erişim; operasyonların sadece kaydedici üzerinde yapılması
- Bütün icra birimlerinin mikrokod kullanmadan donanımsal çalışması
- Yüksek seviyeli dilleri destekleme
- Çok sayıda kaydediciye sahip olması

RISC Mimarisinin Üstünlükleri

- **Hız:** Azaltılmış komut kümesi, kanal ve süperskalar tasarıma izin verildiğinden RISC mimarisi CISC işlemcilerin performansına göre 2 veya 4 katı yüksek performans gösterirler.
- **Basit donanım:** RISC işlemcinin komut kümesi çok basit olduğundan çok az yonga uzayı kullanılır. Ekstra fonksiyonlar, bellek kontrol birimleri veya kayan noktalı aritmetik birimleri de aynı yonga üzerine yarleştirilir.
- **Kısa Tasarım Zamanı:** RISC işlemciler CISC işlemcilere göre daha basit olduğundan daha çabuk tasarlanabilirler.

RISC Mimarisinin Eksiklikleri:

CISC starım stratejisinden RISC tasarım stratejisine yapılan geçiş kendi problemlerinde beraberinde getirmiştir. Donanım mühendisleri kodları CISC işlemcisinden RISC işlemcisine aktarırken anahtar işlemleri göz önünde bulundurmak zorundadırlar.

CISC ve RISC Tabanlı İşlemcilerin Karşılaştırılması

CISC ve RISC tabanlı işlemcilerin karşılaştırılmasında iki önemli faktör farklılıklarını ortaya çıkarmada yeterlidir.

Hız: Genelde RISC çipleri kanal tekniği kullanarak eşit uzunlukta segmentlere bölünmüş komutları çalıştırmaktadır. Kanal tekniği komutları kademeli olarak işler ki bu RISC'in bilgi işlemini CISC'den daha hızlı yapmasını sağlar RISC işlemcisinde tüm komutlar 1 birim uzunlukta olup kanal tekniği ile işlenmektedir. Bu teknikte bazıları hariç komutlar, her bir basamağında aynı işlemin uygulandığı birimlerden geçerler. Kanal teknolojisini açıklamak için herhangi bir komutun işlenmesindeki adımlar ele alınırsa:

Komut kodu ve işlenecek veriler dahil bütün bilgilerin MIB'deki kaydedicilerde olduğu düşünülürse, birinci adımda yapılacak işin kaydedicide bulunan komut kodu çözülür, ikinci adımda üzerinde çalışılacak veri (işlenen) kaydediciden alınıp getirilir, üçüncü adımda veri, komuta göre

Aritmetik ve Mantık Biriminde işleme tabii tutulur ve dördüncü adımda da sonuç kaydediciye yazılacaktır. Böylece bir komutun işlemesi için her bir basamak bir saat çevrimi gerektirirse, dört çevrimle (adımda) gerçekleşmiş olmakta ve bir adım bitmeden diğeri başlayamamaktadır.

Kanal tekniği ile çalışan işlemcilerde birinci adımda komut kodu çözülür, ikinci adımda birinci komutun üzerinde çalışacağı veri (işlenen) kaydediciden alınırken, sıradaki ikinci işlenecek olan komutun kodu çözülür. Üçüncü adımda ilk komutun görevi ALU'da yerine getirilirken, ikinci komutun işleyeceği işlenen alınıp getirilir. Bu anda sıradaki üçüncü komutun kodu çözülür ve işlem böylece devam eder.

Kanal (Pipeline) tekniğinde çevrim zamanın düşmesi için komut kodlarının hızlı çözülmesi gereklidir. RISC mimarisinde tüm komutlar 1 birim uzunlukta oldukları için komut kodunu çözme işlemi kolaylaşır. Sistemde kullanılan kaydedicilerin simetrik bir yapıda olması, derleme işlemini kolaylaştırmaktadır. RISC işlemcilerde belleğe yalnız yükle ve depola komutlarıyla ulaşılır. Bazı eski CISC mimarisinde de olmasına rağmen RISC mimarisinin sabit uzunluktaki basit komutlarla çalışması pipeline sistemini daha iyi kullanmasına sebep olmaktadır. Bu yüzden hesaplama oranlarının birinci öncelik arz ettiği yerlerde iş-istasyonları ve dağıtıcılarda çok tercih edilmektedir.

Transistör sayısı: CISC mimarisinde kullanılan transistor sayısı RISC'e nazaran daha fazladır. Transistör sayısının bir yerde çok olması fazla yerleşim alanı ve ayrıca fazla ısı demektir. Bundan dolayı da fazla ısı üretimi soğutma olayını gündeme getirmektedir. CISC tabanlı Pentium işlemcilerde karışık ısı dağıtıcısı veya soğutma fanlar kullanılmaktadır.

RISC mimarisindeki önemli üstünlüklere karşı bazı mahzurları ortaya çıkmaktadır. RISC mimarisi, CISC'in güçlü komutlarından yoksundur ve aynı işlemi yapmak için daha fazla komut işlenmesini gerektirir. Bundan dolayı da RISC'in bant genişliği artar. Bu sistemde güçlü komutların yokluğu ikinci bir yardımcı işlemciyle ya da işlemci içinde oluşturulacak ayrı bir pipeline bölümüyle giderilebilir. Komut ön-belleğinin kullanılması yüksek komut alıp getirme işlemini azaltmaktadır. RISC mimarisi diğerine nazaran daha kompleks yazılımlara ihtiyaç duyar.

RISC (Hard-wired Control Unit)	**CISC (Microprogrammed Control Unit)**
Hızlı	Nispeten yavaş
Ucuz	Pahalı
Yeniden dizayn zor	Esnek
Daha az komut (instruction)	Daha fazla komut (instruction)
Daha fazla saklayıcı bellek (register)	Daha az saklayıcı bellek (register)

3.2.3. EPIC Mimarisi

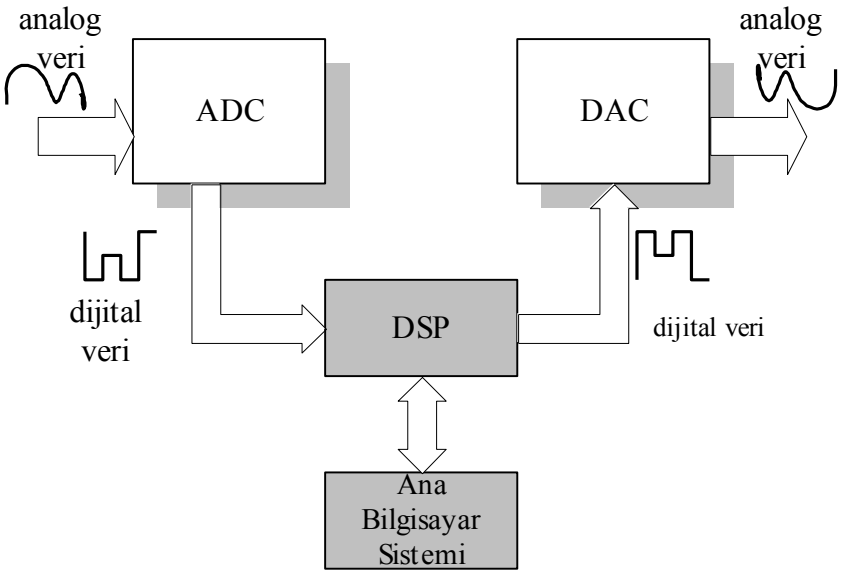
Bu mimari RISC ve CISC mimarisinin üstün yönlerinin bir arada buluştuğu bir mimari türüdür. EPIC mimarisi, işlemcinin hangi komutların paralel çalışabildiğini denetlemesi yerine, EPIC derleyicisinden açık olarak hangi komutların paralel çalışabildiğini bildirmesini ister. Çok uzun komut kelimesi (VLIW) kullanan bilgisayarlar, yazılımın paralelliğine ilişkin kesin bilgi sağlanan mimari örneklerdir. EPIC varolan VLIW mimarisinin dallanma sorunlarını çözmeye çalışarak daha ötesine gitmeyi hedeflemektedir. Derleyici programdaki paralelliği tanımlar ve hangi işlemlerin bir başkasından bağımsız olduğunu belirleyerek donanıma bildirir. EPIC mimarisinin ilk örneği, IA-64 mimarisine dayalı Itanium işlemci ailesidir.

EPIC Mimarisin Üstünlükleri

- Paralel çalıştırma (çevrim başına birden çok komut çalıştırma)
- Tahmin kullanımı
- Spekülasyon kullanımı
- Derleme anında paralelizmi tanıyan derleyiciler
- Büyük bir kaydedici kümesi
- Dallanma tahmini ve bellek gecikmesi problemlerine karşı üstün başarı
- Gelişme ile birlikte eskiye karşı uyumluluk

3.2.4. DSP (Dijital Signal Processing -Dijital Sinyal işleme)

Dijital Signal Processing (Dijital Sinyal işleme) sözcüklerinin bir kısaltmasıdır. 1970'lerin sonlarında mikro-işlemcilerin ortaya çıkmasıyla, DSP kullanımı geniş bir uygulama alanı bulmuştur. Kullanım alanları, cep telefonlarından bilgisayarlara, video çalıcılardan modemlere kadar çok geniş bir alana yayılmaktadır. DSP yongaları, mikro-işlemciler gibi programlanabilir sistemler olup, saniyede milyonlarca işlem gerçekleştirebilir. DSP kartları, üzerlerindeki DSP'ler sayesinde aynı anda bir çok efekt uygulayabilir. Özellikle modemlerde bulunurlar. Çok yüksek hızlarda kayan nokta matematiksel işlemleri yapmak üzere geliştirilmiş bir donanımdır. Diğer birçok şeyin yanı sıra DSP donanımı ses ve görüntü sinyallerinin gerçek zamanlı sıkıştırma ve açma işlemleri için kullanıla bilinir.



Şekil 3.7. DSP sistem ve elemanları

3.3. Bölüm Kaynakları

1. *O. Altınbaşak, 2001. "Mikrodenetleyiciler ve PIC Programlama", Atlaş Yayıncılık, İstanbul.*
2. *N. Gardner, 1998. "PIC Programlama El Kitabı", Bileşim Yayıncılık, İstanbul.*
3. *O. Urhan, M.Kemal Güllü, 2004. "Her Yönüyle PIC16F628", Birsen Yayınevi, İstanbul.*
4. *N. Topaloğlu, S. Görgünoğlu,2003. "Mikroişlemciler ve Mikrodenetleyiciler", Seçkin Yayıncılık, Ankara.*
5. *Y. Bodur, 2001. "Adım Adım PICmicro Programlama",İnfogate.*
6. *Ö.Kalınlı, 2001. "Signal Processing With DSP".*

BÖLÜM 4. BAŞARIM ÖLÇÜTLERİ

Farklı tür bilgisayarların performansını değerlendirebilmek, bu makineler arasında en iyi seçim veya anahtar etmendir. Performans ölçümündeki karışıklık birçok temel etmenden doğar. Komut takımı ve bu komutları tamamlayan donanım önemli ana etmenlerdir. Aynı donanıma ve komut takımına sahip iki bilgisayar bile bellek ve giriş/çıkış örgütlenmesi, ve de işletim sistemleri nedeniyle ya da sadece testlerde farklı iki derleyici kullanıldığından dolayı farklı başarımlar verebilir. Bu etmenlerin başarımı nasıl etkilediğini belirlemek, makinenin belirli yönlerinin tasarımının dayanağı olan ana güdüyü anlamak açısından çok önemlidir.

Yolcu uçaklarıyla ilgili bir örnek verelim. Aşağıda mevcut uçaklarla ilgili bilgiler verilmiştir. Buna göre; 5000 yolcuyu New York'tan Paris'e (1500 km) taşımamız gerektiğine göre hangi uçağı kullanmalıyız?

Uçak	P: Yolcu kapasitesi	R: Uçuş menzili (km)	S: Uçuş hızı (km/saat)	(P*S) Yolcu gönderme hızı
Boeing 737-100	101	1000	1000	101000
Boeing 777	375	7400	1000	375000
Boeing 747	470	6650	1000	460600
BAC/Sud Concorde	132	6400	2200	290400
Douglas DC-8-50	146	14000	880	128480

Tablo 4.1. Uçaklar ve özellikleri

Boeing 737-100 uçuş menzili New York'tan Paris'e uçmaya yetmeyeceğinden daha ilk başta listeden elenir. Ardından, geriye kalanlar arasında Concorde en hızlı olarak görülmektedir. Ancak uçağın sadece hızlı olması yeterli değildir. Boeing 747 daha yavaş olmasına karşın bir Concorde un taşıyabileceğinden 3 kat daha fazla yolcu taşıyabiliyor. Uçakların performansları için daha iyi bir ölçüt uçakların yolcu taşıma hızı olabilir. Yolcu sayısının uçağın hızıyla çarpımından çıkan sayı yolcu taşıma hızıdır. Bu durumda 747 nin 5000 yolcuyu taşımada daha başarılı olduğunu görürüz, çünkü onun yolcu taşıma hızı Concorde dan daha yüksektir. Diğer taraftan, eğer toplam yolcu sayısı 132 den az olursa Concorde elbetteki Boeing 747 den daha iyidir. Çünkü onları 747 den neredeyse iki kat hızlı taşıyacaktır. Yani performans büyük oranda yapılacak işe bağlıdır.

4.1. Başarım Tanımı

Bir bilgisayarın diğerinden daha iyi başarıma sahip olduğunu söylemekle, tipik uygulama programlarımız açısından o bilgisayarın birim sürede diğerinden daha çok iş bitirebildiğini kastederiz.

Zaman paylaşımlı çok-kullanıcılı çok- görevli bir bilgisayarda, bir programın başlangıcından bitişine kadar geçen toplam zamana toplam yürütme süresi denir. Genellikle giriş/çıkış ile programımızın işlemesi için geçen süre ayrı ayrı sayılır. Bunlar işimizin CPU süresi ve Giriş/ Çıkış işlem süresi olarak adlandırılır. Zaman paylaşımlı anabilgisayarlarda diğer kullanıcıların işleri arasında çalışan programın çalışma süresi CPU süresinden daha uzundur. Kullanıcıları genelde bu çalışma süresi ilgilendirirken, bilgisayar merkezinin yöneticisi bilgisayarın toplam iş bitirme hızıyla (throughput) ilgilenir.

Programların CPU sürelerini azaltmak için çeşitli yöntemler vardır. Bunlardan akla ilk gelen bilgisayarı aynı tip daha hızlı sürümüyle değiştirmektir. Bu yöntem başarımı kısmen arttırır. Belli bir görevde , X bilgisayarının başarımı temel olarak programın çalışma zamanıyla ters orantılıdır.

$$Xin\ Başarımı = \frac{1}{Xin\ Çalışma\ süresi}$$

Bu da, X ve Y bilgisayarlarının başarımı çalışma zamanıyla ters orantılıdır demektir.

$$Y'nin\ Çalışma\ Süresi > X'in\ Çalışma\ Süresi$$

ise

$$X'in\ Başarımı > Y'nin\ Başarımı$$

demektir. Nicel olarak,

$$\frac{X\ in\ Başarımı}{Y\ nin\ Başarımı} = \frac{Y\ nin\ Çalışma\ Süresi}{X\ in\ Çalışma\ Süresi} = n$$

ise X in Y den n kat hızlı olduğu söylenir.

4.2. Ölçme Koşulları ve Ölçme Birimleri

Çok görevli ve çok kullanıcılı bir bilgisayar ortamında yürütme süresi ve belli bir iş için harcanan işlem süresi farklı kavramlardır.

- Programın başlatılmasına bitişine kadarki zamana toplam yürütme süresi, yanıt zamanı, geçen süre yada duvar süresi denir.
- Program işlemesinde CPU tarafından harcanan zaman dilimlerinin toplamına CPU yürütme süresi yada basitçe CPU süresi denir.
- CPU süresi daha da ayrışarak program CPU süresi ve sistem CPU süresine bölünür. Sistem CPU süresi içinde giriş/çıkış, disk erişimi ve benzeri diğer çeşitli sistem görevleri yapılır. Program CPU zamanı ise yalnızca program kodunun yürütülmesi için geçen net süredir.

Zaman genellikle saniye(s) birimiyle ölçülür. Ancak saat dönüş süresi, yani bilgisayarın peryodu çoğunlukla nanosaniye (nano=1/1 000 000 000) kullanılarak ölçülür. Genelde bilgisayarların hızları verirken saat hızı (=1/saat-dönüşü) tercih edilir. Saat hızının birimi Hertz (Hz)

dir. Hertz saniyedeki dönüş sayısına eşittir. Daha hızlı saatler için Kilo-Hertz, Mega-Hertz yada Giga-Hertz terimleri kullanılır.

Tablo 4.2. Zaman Birimleri

Zaman Birimleri	**Saniye**	**Mili-saniye**	**Mikro-saniye**	**Nano-saniye**
Kısaltması	s	Ms	μs	ns
Saniye eşdeğeri	1	0.001	0.000 001	0.000 000 001

Tablo 4.3. Frekans birimleri

Frekans Birimleri	**Hertz**	**Kilo- Hertz**	**Mega- Hertz**	**Giga- Hertz**
Kısaltması	Hz	KHz	MHz	GHz
Saniyedeki dönüş	1	1000	1 000 000	1 000 000 000

Bilgisayarların başarımlarını karşılaştırırken, gerçekte kullanılacak uygulama programlarının iş-bitirme hızı son derece önemlidir. Bir programın CPU yürütme süresini belirleyen temel ifade;

CPU-yürütme-süresi= CPU-saat-dönüş-sayısı × Saat dönüş süresi;

Biçimindedir.

CPU-saat-dönüş-sayısı ise;

CPU-saat-dönüş-sayısı = komut sayısı × komut başına ortalama dönüş sayısı

Komut başına ortalama dönüş sayısı genellikle CPI (cycle-per- instruction) diye adlandırılır.

ÖRNEK 4.1: A ve B aynı komut takımına sahip iki makine olsun. Herhangi bir program için A'nın saat dönüşü 10ns ve CPI 'si 2.0 ölçülmüş, aynı program için B'nin saat dönüşü 20ns ve CPI'si 1.2 ölçülmüştür. Bu program açısından hangi makine kaç kat hızlıdır.?

Çözüm 4.1: Programdaki komut sayısının I olduğunu varsayalım. Bu durumda;

CPU-süresi-A = CPU-saat-dönüşü-sayısı-A × saat- dönüş süresi A

$= I \times 2.0 \times 10\ ns = 20\ I\ ns$

CPU süresi B $= I \times 1.2 \times 20\ ns = 24\ I\ ns$

CPU-süresi-A < CPU süresi B , o halde A daha hızlıdır.

$$\frac{Başarım\ A}{Başarım\ B} = \frac{Çalışma\ Süresi\ \ B}{Çalışma\ Süresi\ \ A} = n$$

$n = 24 \times I\ ns / 20 \times I\ ns = 1.2$

A makinesi B den 1.2 kat daha hızlıdır.

4.3. Yaygın Kullanılan Yanıltıcı Başarım Ölçütleri

MIPS ve MFLOPS, sistem başarımını karşılaştırmak için sık kullanılan başarım ölçütleridir. Bu iki başarım ölçütü birçok durumda yanıltıcı olabilir.

4.3.1. MIPS Başarım Ölçümü

MIPS saniyede milyon komut için kısaltmadır. Bir programda,

$$MIPS = \frac{Komut\ Sayısı}{Yürütme\ Süresi \times 10^6} = \frac{Komut\ Sayısı}{CPU\ -\ saat\ -\ dönüş\ -\ sayısı \times saat\ -\ dönüş\ süresi \times 10^6}$$

$$= \frac{Komut\ sayısı \times saat\ hızı}{Komut\ sayısı \times CPI \times 10^6}$$

burada *CPU saat dönüşü sayısı = komut sayısı × CPI* olduğundan

$$MIPS = \frac{Saat\ hızı}{CPI \times 10^6} \qquad (doğal\ MIPS)$$

$$Çalışma\ Süresi = \frac{Komut\ sayısı \times CPI}{Saat\ hızı} = \frac{Komut\ sayısı}{Saat\ hızı \times 10^6\ /\ CPI \times 10^6}$$

$$Çalışma\ Süresi = \frac{Komut\ sayısı}{MIPS \times 10^6}$$

bu eşitliğe göre hızlı makinenin MIPS değeri yüksektir diyebiliriz.

MIPS Ölçümünü Kullanmanın Sakıncaları

- Aynı iş kullanılan komut sayıları farklı olacağından farklı komut takımlarına sahip bilgisayarları MIPS kullanarak karşılaştıramayız.
- Aynı bilgisayar da çalıştırılan farklı programlar farklı MIPS değerleri veriri. Bir makinenin tek bir MIPS değeri olamaz.

Bazı durumlarda MIPS gerçek performansa ters yönde değişebilir.

4.3.2.MFLOPS ile Başarım Ölçümü

MFLOPS saniyede milyon kayan noktalı işlem anlamına gelir. Her zaman "megaflops" diye okunur.

$$MFLOPS = \frac{Bir\ programdaki\ kayan\ noktalı\ işlemler\ sayısı}{Yürütme\ süresi \times 10^6}$$

MFLOPS programa bağlıdır. Komutlar yerine aritmetik işlemlerin üzerinde tanımlandığından, MFLOPS farklı makineleri karşılaştırmada daha iyi bir ölçüt olma eğilimindedir. Ancak, farklı makinelerin kayan noktalı işlem takımları birbirine benzemez ve gerçekte aynı iş için gereken kayan noktalı işlem sayısı her makinede farklı olabilir.

4.3.3.Başarım Değerlendirme Programlarının Seçimi

MIPS ve MFLOPS yanıltıcı başarım ölçütleridir. Bir bilgisayarın başarımını ölçmek için,"benchmark"(karşılaştırma noktası" adı verilen bir grup karşılaştırma programını kullanarak değerlendiririz.

- Karşılaştırma programları kullanıcının gerçek iş yükünün vereceği başarımı tahmin edecek iş yükünü oluşturur.
- En iyi karşılaştırma programları gerçek uygulamalardır, ancak bunu elde etmek zordur.

Seçilen karşılaştırma programları gerçek çalışma ortamını yansıtmalıdır.Örneğin;tipik bazı mühendislik yada bilimsel uygulama mühendis kullanıcıların iş yükünü yansıtabilir.Yazılım geliştirenlerin iş yükü ise, çoğunlukla derleyicidir, belge işleme sistemleri ,vb. –den oluşur.

Benchmark sonuçları rapor edilirken, makinelerin karşılaştırma ölçümleri ile birlikte şu bilgilerde listelenmelidir.

- İşletim sisteminin sürümü
- Kullanılan derleyici
- Programa uygulanan girdiler
- Makine yapılanışı(bellek, giriş/çıkış hızı, vs)

Daha yüksek başarım sonuçları elde edilen makine sisteminin belirlenmesinde;

Donanım	
Model no	Powerstation 550
CPU	41.67 MHz POWER 4164
FPU	Tümleşik
CPU sayısı	1
Önbellek Boyutu	64k veri, 8k komut
Bellek	64 Mb
Disk alt sistemi	2-400 SCSI
İletişim ağı arayüzü	Yok
Yazılım	
O/S tipi	AIX v3.1.5
Derleyici sürümü	AIX XL C/6000 ver 1.1.5 AIX XL Fortran ver 2.2
Diğer yazılım	Yok
Dosya sistemi tipi	AIX
Bellenim seviyesi	YOK
Sistem	
Uyum parametreleri	Yok
Art alan yükü	Yok
Sitem durumu	Çok kullanıcı (tek kullanıcı login)

Tablo 4.3. Daha yüksek başarım sonucu elde edilen makine sisteminin betimlenmesi

SONUÇ

- Doğru başarım ölçüsü üç parametreyi:komut sayısı,CPI, ve saat hızı-nı şu şekilde içermelidir

$$Yürütme\ Süresi = \frac{Komut\ sayısı \times CPI}{Saat\ hızı}$$

- Bir tasarım farklı yönlerinin bu anahtar anahtar parametrelerin her birini nasıl etkilediğini anlamamız gerekir: Örneğin,

  -Komut takımı tasarımı komut sayısını nasıl etkiler,

  -Ardışık düzen ve bellek sistemleri CPI değerini nasıl etkiler,

  -Saat hızı teknoloji ve organizasyona nasıl bağlıdır.

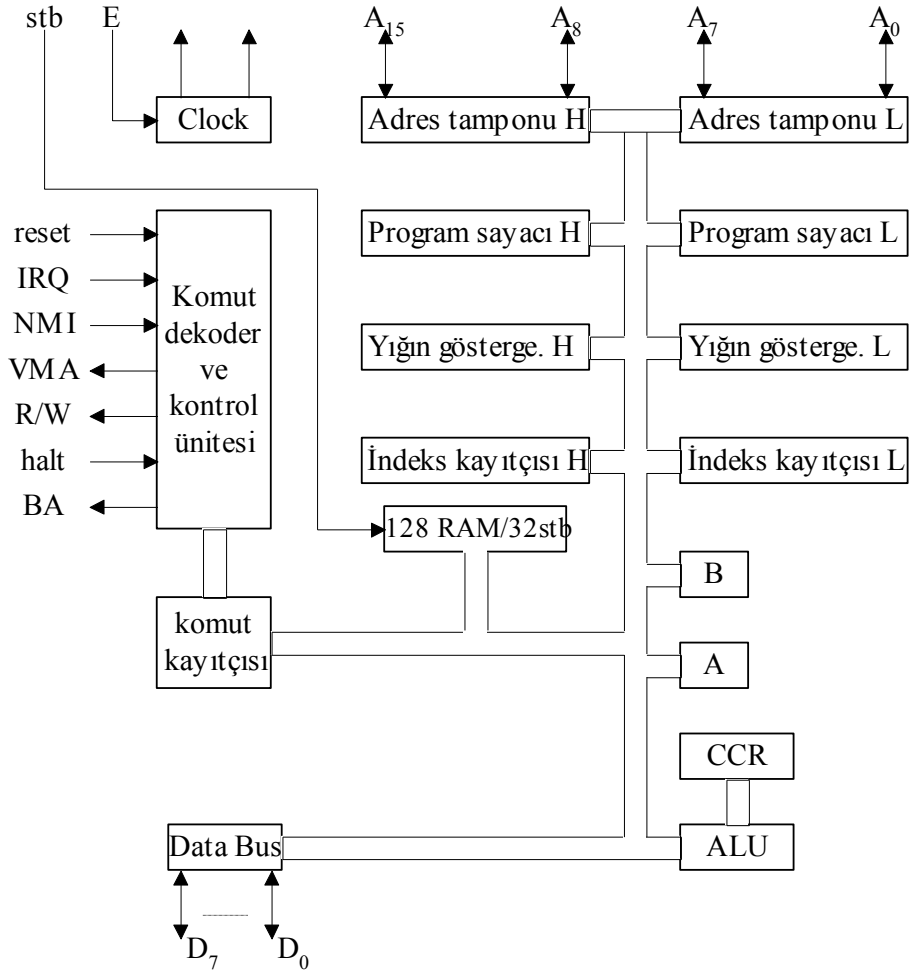
- Sadece başarıma bakmamız yetmez, maliyeti de düşünmemiz gerekir. Maliyet şunları kapsar:
  - Parça maliyeti
  - Makineyi yapacak iş gücü
  - Araştırma ve geliştirme giderleri
  - Satış, pazarlama, kar, vs.

4.4. Bölüm Kaynakları

1. *M. Bodur, "RISC Donanımına GİRİŞ", Bileşim Yayınevi*

BÖLÜM 5. VON NEUMAN VE CISC MİMARİLİ MİKROİŞLEMCİ

5.1. MC6802 Mikroişlemcisinin Yapısı Ve Kayıtçıları



Şekil-5.1. 6802 mikroişlemcisinin yapısı.

Mikroişlemci resetlendiğinde veya enerji verildiğinde adres çıkışı ilk anda; $A_0 = 0$, A_1.....$A_{15} = 1$ olur. Yani FFFE dir. Bu adresteki bilgi otomotikmen program sayacına (PC) yüklenir ve program sayacındaki bu yeni adres değerinden itibaren mikroişlemci çalışmasına başlar. Yani; [FFFE] = 80 ve [FFFF] = 00 varsa bu değerler program sayacına yüklenir PC = 8000. Artık [8000] adresindeki programa göre mikroişlemci çalışır.

Program sayacı (program counter, PC) : Adres ucu kadar bite (16 bit) sahip kayıtçıdır. O anda çalışacak olan komutun adresini üzerinde bulundurur. Komut çalıştırıldıktan sonra değerini bir artırır.

Komut kayıtçısı (instruction register, Ir) : O anda çalışan komutu üzerinde bulundurur.

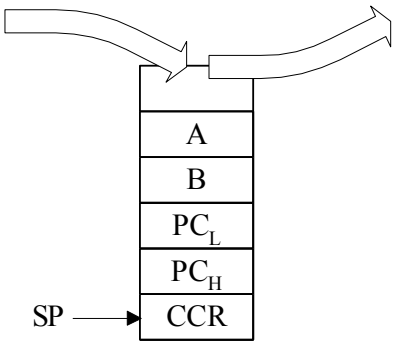
Komut kod çözücü (instruction decoder) : Komut kayıtçısından gelen bilgileri, kontrol sinyalleri oluşturacak şekilde kodlar.

Akümülatör (accumulator, birikeç, A, B) : A ve B olmak üzere iki tanedir. Data ucu kadar bite sahiptir (8 bit). ALU tarafından kullanılırlar. Genelde o andaki dataları veya işlem sonuçlarını üzerinde bulundururlar.

İndis kayıtçısı (index register, X) : 16 bitliktir. Kullanılacak gerçek hafıza yerini belirlemek için bu kayıtçı içindeki değer, komutla belirtilen adrese eklenir.

Yığın göstergeci (stack pointer, SP) : 16 bitliktir. Hafızadaki herhangi bir hücre adresini üzerinde bulundurur. Herhangi bir dallanma alt programlara gitme ve kesme istekleri anında mikroişlemcinin o andaki bilgilerini dönüş anında kullanmak üzere saklamak gerekir. Bunun içinde geçici olarak yığın göstergecinin RAM üzerinde göstermiş olduğu adresten geriye doğru bir data yığını oluşturulur. SP ise bu data yığınının oluşturulacağı adres başlangıcını üzerinde tutar. Yığına son atılan bilgi ilk alınır. Yığının kapasitesine bağlı olarak içi içe dallanmalar yapılabilir. Eğer yığının kapasitesi yetersiz ise yığın taşması (stack overflow) problemi ortaya çıkar.

- Yığından bir okuma/yazma yapılacaksa, SP'nin işaret etmiş olduğu hafıza hücresinden okunur/yazılır.
- SP'nin değeri mikroişlemci tarafından otomatik olarak arttırılır ya da azaltılır.
- Yığın türleri:
  - LIFO (Last-In First-Out): Yığına son atılan bilgi ilk alınır.
  - FIFO (First-In, First-Out) : Yığına ilk atılan bilgi ilk alınır.
- Bir PUSH komutuyla veri, yığına atılırken, PULL komutu ile veri yığından alınır.



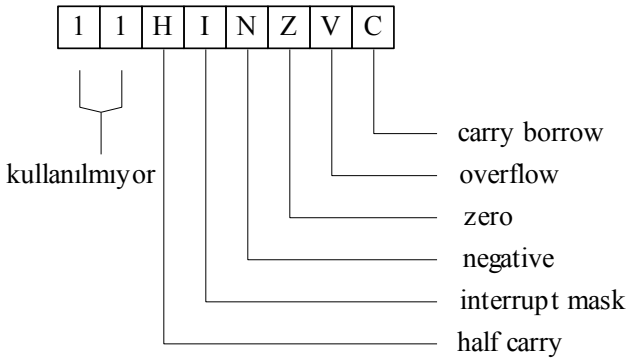
RAM üzerinde

Şekil-5.2. Yığın

- Yığın çok düzeyli kesmelerin kolayca gerçeklenmesini, sınırsız sayıda alt programın iç içe geçirilebilmesini ve birçok veri işleme türlerinin basitleştirilmesini sağlar.
- Mikroişlemcinin ana programdan alt programa gittiği zaman ana programa geri döneceği adresi sakladığı adres gözünün adresini içerir. Ana programdan alt programa gidildiği zaman PC' de o anda ana program komut satırının adresi vardır.

Durum kod kayıtçısı (condition code register, CCR) : ALU ile birlikte çalışır. Bayrak kaydedicisi, bütün mikroişlemcilerde olduğu gibi, tipine bağlı olarak 8-bit, 16-bit ve 32-bit olmak üzere, bir işlemin sonunda sonucun ne olduğunu kaydedici bitlerine yansıtan bir bellek hücresini oluşturur. Bu kaydediciye bayrak denmesinin sebebi, karar vermeye dayalı komutların yürütülmesinde sonuca göre daha sonra ne yapılacağını bit değişimiyle bu kaydedicinin 1-bitlik hücrelerine yansıtmasıdır. Kaydedicideki bitlerin mantıksal 1 olması bayrak kalktı, mantıksal 0 olması bayrak indi anlamındadır. Karşılaştırma ve aritmetik komutların çoğu bayraklara etki eder.

MC6802 Mikroişlemcisinin CCR kayıtçısı 8-bitlik olup, üzerinde şu bilgiler bulunur:



Şekil-5.3. Durum kod kayıtçısı bayrakları.

Z-biti : İşlem sonucu sıfır ise bu bit lojik-1 olur.

N-biti : İşlem sonucu negatif ise bu bit lojik-1 olur.

H-yarım elde biti : Yapılan toplam işlemi sonucunda elde biti oluşmuş ise bu bit lojik-1 olur.

V-taşma biti : Eğer bir elde biti varsa ve daha sonra yapılan işlem sonucunda tekrar elde biti oluşmuş ise bu bit lojik-1 olur.

C- borç elde biti : Bir çıkarma işleminde çıkan sayı çıkarılan sayıdan büyük ise borç alınır. Bu durumda bu bit lojik-1 olur.

I-kesme biti : Bu bit lojik-o ise gelen IRQ kesme isteklerine izin verilir. Eğer bu bit lojik-1 ise gelen IRQ kesme isteklerine izin verilmez.

6802 Mikroişlemcisinin Kontrol Sinyalleri

- IRQ (interrupt request) : kesme isteği
- Reset : al baştan
- Halt : duraklatma kesmesi
- NMI (non-maskable interrupt) : maskelenmeyen kesme
- R/W : oku/yaz
- VMA (valid memory address) : geçerli adres ucu

- 3 durumlu kontrol
- BA (BUS available) : yol kullanılabilir.
- DATA BUS enable

Halt : Bu uç lojik-0 olduğunda 6802 son komutunu (en son yaptığı işlemi) tamamlar ve çalışmasını durdurur. Bu durumda adres yolu bir adresin komutunu gösterir. BA lojik-1 olur. VMA ucu ise lojik-0 olur. Eğer kesme (halt) işlemi yapılmayacaksa, halt ucu +5 volta bağlanmalıdır.

R/W : Lojik-0 ise hafızaya yaz. Lojik-1 ise hafızadan oku manasındadır. Bir çıkış ucu olup, hafıza ve giriş/çıkış ünitelerine yazmak ve okumak için kullanılır.

VMA : Adres hatları üzerindeki bilginin adres bilgisi olup olmadığını belirlemeye yarar. Buda bir çıkış ucu olup gerekli çevre birimlerle bağlantısı yapılmalıdır. Çevre birimler; hafıza, giriş/çıkış ünitesi ...v.s. olabilir.

BA : Bu çıkış data ve adres yollarının mikroişlemci dışındaki kullanıcılar için kullanmaya uygun olduğunu belirler. Örneğin halt kesmesi gelince o andaki adresteki bilgilerin kullanılabileceğini gösterir.

Reset : Bu uç lojik-0 yapıldığında program FFFE ve FFFF adresindeki yazılı olan adrese dallanır ve FFFE ile FFFF nin gösterdiği adres mikroişlemci programının başlangıç adresidir. Yani mikroişlemci her çalışmasında bu adrese göre çalışacaktır.

NMI : Bu uç lojik-0 olunca mikroişlemcinin o andaki bilgileri yığın göstergeci vasıtasıyla saklanır. NMI lojik-0 olunca kesme anında yapılması gereken işler için FFFC, FFFD adresinde belirtilen adresteki programa dallanır. O adresteki program bitince tekrar çalışmasına kaldığı yerden devam eder. Yani uca gelen kesme beklemeden devreye girer.

IRQ : CCR kayıtçısında belirtilen kesme (I) biti ile denetlenir. Eğer I biti sıfır ise gelen kesme isteğine cevap verilir. Bu bit lojik-1 ise kesme isteği geçersizdir. Bir kesme başlamışsa bir diğer kesmeye izin vermemek için bu bit lojik-1 yapılmalıdır. Mikroişlemcinin gerekli bilgileri yığın göstergeci yardımıyla saklanır ve sonra FFF8, FFF9 adreslerinde yazılı olan adresteki programa dallanır.

SWI : Diğer kesmelerin aksine bir yazılım kesmesidir. Bu kesme geldiğinde FFFA, FFFB adreslerinde belirtilen adresteki programı çalıştırır. Çoğu mikrobilgisayarda bu kesme geldiğinde monitörü durdurucu ve aynı zamanda mikrobilgisayarı duraklatma işlevini yapan bir program çalıştırılır.

FFF8	IRQ	H	H: yüksek değerlikli byte
FFF9	IRQ	L	
FFFA	SWI	H	L: düşük değerlikli byte
FFFB	SWI	L	
FFFC	NMI	H	
FFFD	NMI	L	
FFFE	RESTART	H	
FFFF	RESTART	L	

Şekil-5.4. Kontrol sinyallerinin adres yerleşimi.

5.2. MC6802 İle Gerçekleştirilmiş Mikrobilgisayar

A) MİNİMUM 6802 MİKROBİLGİSAYAR DEVRESİ

"Minimum mikrobilgisayar = CPU + Hafıza + Giriş/Çıkış Ünitesi" den oluşur.

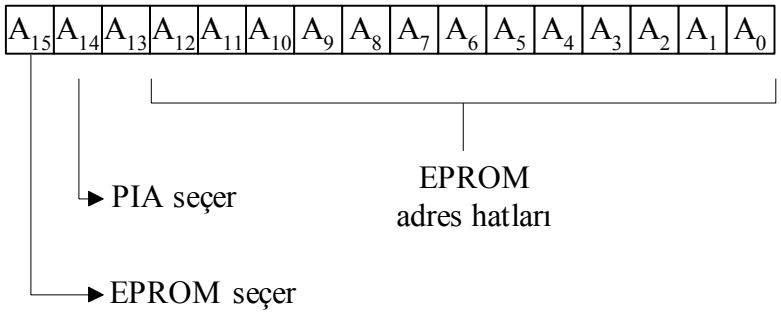
Örnek:

CPU → 6802-1Mhz,

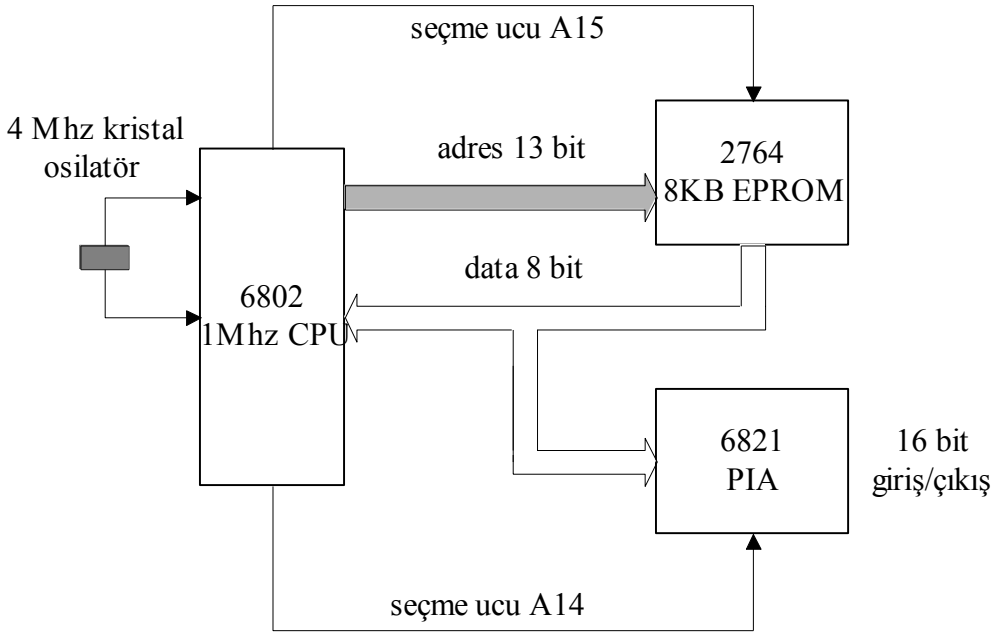
Hafıza → 2764-8KB Eprom

Giriş/Çıkış → 6821 PIA olsun.

Bunun blok şemasını çizmek istersek, CPU nun 16 bit adres hatlarını aşağıdaki gibi dağıtmamız mümkündür.



Şekil-5.5. CPU adres hatlarının dağılımı.



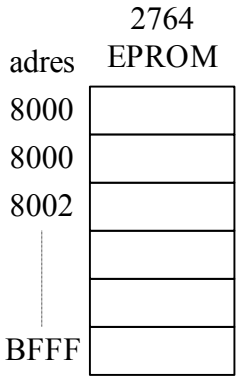
Şekil-5.6. Minimum mikrobilgisayar blok diyagramı.

- Giriş / çıkış ünitesi olarak PIA (paralel giriş/çıkış) kullanılmıştır. Kullanılan MC6821 PIA nın 2 adet 8 bitlik giriş/çıkış portu vardır.

- MC6802 mikroişlemcisinin 16 adres ucu vardır. Bunun 13 adetini 2764 Epromu için, 2 tanesi de eprom ve PIA yı seçme işlemi için kullanıldı.
  $A_{15} = 1$, $A_{14} = 0$ ise EPROM seçilir.
  $A_{15} = 0$, $A_{14} = 1$ ise PIA seçilir.

- Seçme uçlarında dikkat edilmesi gereken, herhangi bir anda sadece bir elemanın seçilmesidir.

- EPROM un herhangi bir hafıza bölgesinden bilgi okuyacaksak, A_{15} adres ucunu aktif yapacak şekilde adres vermeliyiz.



Şekil-7.3. EPROM seçimi ve adresleri.

- PIA üzerinden giriş yada çıkış işlemi yapacaksak, A_{14} uçunu aktif yapacak şekilde bir adres vermeliyiz. Örneğin; 4000.

- MC6802 mikroişlemcisinin içerisinde 128 byte lik bir RAM vardır. Bu RAM 0000_H adresine yerleştirilmiştir. Yani 128 bytelik RAM ın adres aralığı 0000-007F dir ve 7 adet adres ucu vardır. Bu RAM ın ilk 32 bytelik kısmı stand-by (stb) özelliği vardır. Bu özellik; mikroişlemcinin enerjisi kesilse bile dışardan bir pil ...v.s. ile stb li kısım beslenerek üzerindeki bilginin silinmesini engeller. Stb gerilimi 4.5 V + %10 dur.

- MC6802 mikroişlemcisinin içinde bir clock puls ünitesi vardır. Bu ünitenin çalışması için uçlarına bir kristal osilatör bağlamak gerekir. Bağlanacak olan bu osilatörün frekansı mikroişlemcinin frekansının 4 katı olmalıdır. MC6802-1 Mhz için 4Mhz lik, MC6802-2 Mhz için 8Mhz lik bir kristal osilatör bağlanmalıdır.

- MC6802 mikroişlemcisi 8 bit data bus, 16 bit adres ucu ile 74 assembly komuta sahip bir mikroişlemcidir.

5.3. Assembly (Birleştirici) Dil Kuralları

Bir assembly satırı su şekildedir :

Etiket Alanı	Komut Alanı	Veri Alanı	Açıklayıcı Bilgi

Etiket Alanı

Etiket kullanılmıyorsa, komut bir sütün içerden başlanmalıdır

İlk sütunda * varsa bundan sonraki bilgilerin açılama olduğunu gösterir.

Eğer etiket varsa şu kurallara uyulmalıdır :

Etiket 1 ile 6 karakter uzunluğa sahip olabilir.

İlk karakter sayı ve rakam olmamalıdır

İlk sütundan başlanılmalıdır.

Etiket program boyunca aynı isimden bir tane olmalıdır.

Bir etiket şu durumlarda kullanılır.

Herhangi bir şartlı dallanma komutu ile gidilecek yeri belirlemek.

Herhangi bir şartsız dallanma komutu ile gidilecek yeri belirlemek.

Herhangi bir alt programa gitmek için.

Komut Alanı

Bu alanda ilgili mikroişlemcinin assembly komutları bulunabilir.

Komut (assembly)	Operasyonel kod (op-code)	İkilik sistemde
CLR A	4F	0100 1111
CLR B	5F	0101 1111

MC6802 mikroişlemcisinin 74 komutu bulunmaktadır. Bu komutları dört guruba ayırabiliriz.

a- 8 bitlik kayıtçılar ile yapılan işlem komutları

İki işlemli aritmetik : ABA, ADD,...

Tek işlemli aritmetik : CLR A, INC B, DEC A, ...

Karşılaştırma ve test etme : CMP, BIT A, ...

Kaydırma ve döndürme : ASL, ROR, ...

Mantıksal fonksiyonlar : EOR, AND, COM, ...

Yükleme ve depolama işlemleri : LDA, STA, LDX, STS,

Transfer : TAB, TBA,TXS, ...

b- Atlama ve dallanma komutları

Şartlı dallanma : BNE, BEQ,....

Şartsız dallanma : BRA, JMP, ...

Alt programlara gitme : BSR, JSR, ...

Kesme işlemlerinde belirtilen yerlere gitme : WAI, ...

c- İndis kayıtçısı ve yığın göstericisi kontrolü yapan komutlar

indis kayıtçısı ile ilgili işlemler : LDX, INX, STX, DEX, CPX, ...

Yığın göstericisi ile ilgili işlemler : STS, LDS, INS, ...

Transfer işlemleri : TSX, TXS, ...

d- Durum kod kayıtçısı kontrolü yapan komutlar

Bit kontrolü : CLI, SEI, CLC, ...

Byte kontrolü : TAP, TPA,

Veri Alanı

Adresleme modlarına göre bilginin girildiği alanlardır. Bu bölgedeki bilgilere göre komutlar 1, 2 veya 3 bytelik olmaktadır. Eğer bir komut 2 veya 3 byteden oluşmakta ise 2. veya 2. ve 3. byteler bir işlem, bir adres veya adres elde etmekte kullanılacak bilgiyi içerir. Kısaca işlemler ve karakterler veya karakter dizileri bu bölgede bulunabilir. Bu bölgede bulunan bilgiler assembly derleyicisi tarafından şu şekilde anlaşılır :

Sayıların Temsili	Derleyici Tarafından Anlaşılması
sayı	Desimal –10
$sayı sayıH	Hekzadesimal – 16
@sayı sayıO sayıQ	Oktal – 8
%sayı sayıB	Binary – 2

İşaretler	Anlamı
# (diyes)	Kendisinden sonra gelenin data olduğunu gösterir ve immediate adreslemede kullanılır.
+, -, *, /	Normal matematiksel işlemler için kullanılır (üst seviyeli programlama dillerinde : C, pascal, fortran, ...v.s.).

Örnek : 60_H ile 61_H adresindeki bilgileri toplayıp 62_H adresine yazan bir assembly program yazınız.

Assembly Dil	Operasyonel kod	Makine dili
LDA A 60_H	96	1001 0110
	60	0110 0000
ADD A 61_H	9B	1001 1011
	61	0110 0001
STA A 62_H	97	1001 0111
	62	0110 0010

Programın makine kodundaki halini anlamak oldukça zordur. Hekzadesimal haldeki operasyonel kodlarında kullanıcı tarafından anlaşılması zordur. Bu nedenlerden dolayı daha anlaşılır olan assembly dili kullanılır. Daha sonra program EPROM a veya mikroişlemciye verilirken operasyonel kodlara çevrilerek verilir.

5.4. MC6802 Mikroişlemcisinin Komut Kümesi ve Adresleme Modları

A) KOMUT KÜMESİ

6802 mikroişlemcisinin komut kümesi bir, iki veya üç bytelik komutlardan oluşur. Bir komutun uzunluğu komuta ve adresleme çeşidine bağlı olup, ilk (veya tek) byte komutu ve kullanılan adresleme çeşidini belirlemeye yeterlidir. 6802 mikroişlemcisinin 74 komutu bütün

geçerli adresleme çeşitleri için onaltılık tabanda operasyonel kodları EK-1,2 deki tablolar da verilmiştir. Kullanılabilecek 256 (00...FF) onaltılık sayıdan 197 si geçerli birer makine kodu olduğu, 56 sının ise kullanılmadığı verilen tablodan görülebilir.

B) ADRESLEME MODLARI

LDA A 60_H gibi bir assembly satırında; 60_H bilgisimi A akümülatörüne yüklenecek, yoksa 60_H adresindeki bilgimi A akümülatörüne yüklenecek? İşte bunun gibi durumları ayırt etmek için adresleme modlarına gerek vardır.

6802 mikroişlemcisinde 7 adresleme modu kullanılabilmektedir.

- Immediate (hazır, hemen, derhal, anında, ivedi, öncel) adresleme
- Relative (bağıl, dolaylı, göreceli) adresleme
- Inherent (doğal, anlaşılır, içerilmiş) adresleme
- Indexed (indisli, sıralı) adresleme
- Akümülatör adresleme (Anlaşılır adreslemenin özel bir durumudur)
- Extended (genişletilmiş, mutlak) adresleme
- Direct (doğrudan) adresleme

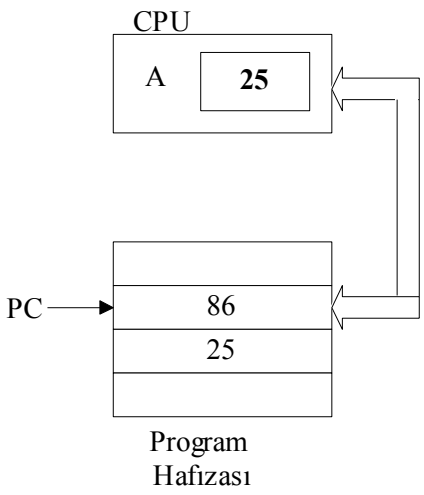
Bu adresleme modları mikroişlemciler için temelde aynı olmasına rağmen bazı değişiklikler gösterir.

1. Anında adresleme

Bu yöntemde işlenecek olan bilgi, komutun 2. byte dında yer alır. Anında adreslemeyi assembly dil yazılımında belirlemek için verinin önüne '#' işareti konur.

Örnek :

Assembly	op-code	yaptığı iş
LDA A #$25	86 25	A = 25_H
LDX #$1000	CE 10 00	X = 1000_H
LDA A #45	86 45	A = 45

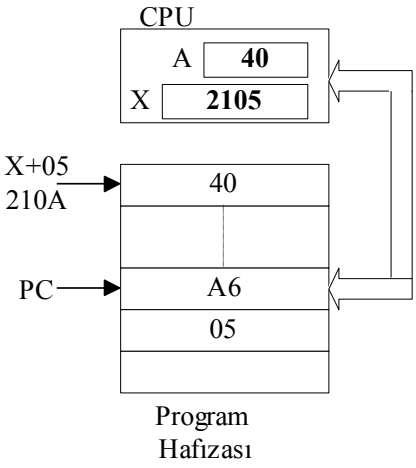


2. İndisli adresleme

İndis kayıtçısı kullanılarak adreslemenin yapılmasından dolayı, bu yönteme indisli adresleme yöntemi denilmiştir. Komuttan sonra gelen sayı indis kayıtçısındaki sayıya eklenerek, gerçek data adresi belirlenir. Belirlenen bu adresten data okunur veya yazılır. İndisli adresleme yöntemini belirtmek için komut ve datadan sonra '**x**' yazılır.

Örnek :

Assembly	op-code	yaptığı iş
LDA A $05,X	A6 05	A = [X+05]

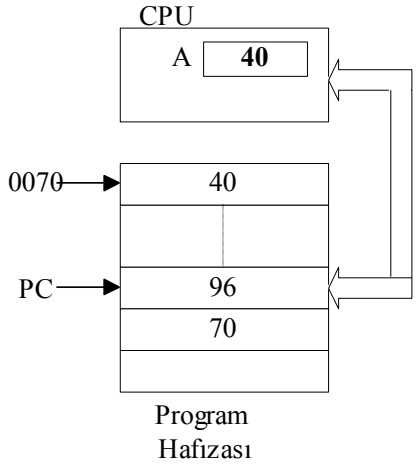


3. Doğrudan adresleme

Doğrudan adresleme yönteminde komutun operasyonel kodundan sonra işlenecek olan verinin bulunduğu adres yazılır. Bilindiği üzere 16 bitlik adresler 0000....FFFF arasındadır. Doğrudan adreslemede adres 8 bit kullanılarak, 8 bitlik adreslere ulaşılırken fazladan yer kaplamamak için 0000....00FF arasındaki adresler, 00....FF şeklinde kullanılmaktadır. Yani doğrudan adreslemede komuttan sonra gelen adres değeri bir bytedir.

Örnek :

Assembly	op-code	yaptığı iş
LDA A $70	96 70	A = [0070]
LDX $70	DE 70	X = [0070,0071]

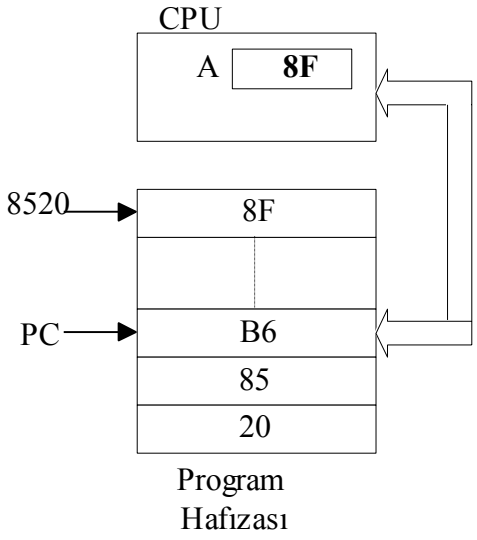


4. Genişletilmiş Adresleme

Bu adresleme, doğrudan adreslemenin genişletilmiş bir şekli olup $0000.....$FFFF arasında tüm durumlara erişilmesini sağlar. Bu durumlar iki bytelik olduğu için genişletilmiş adresleme komutları 3 byteden oluşur.

Örnek :

Assembly	op-code	yaptığı iş
LDA A $45	96 45	A = [0045]
LDA A $0045	B6 0045	A = [0045]
LDX $8520	FE 8520	X = [8520,8521]
LDA A $8520	B6 85220	A = [8520]



5. Anlaşılır adresleme

Bu adreslemede işlenecek olan veri (data, bilgi) komutun kendisi ile birlikte verilir. Böylece işlenecek olan bilgi herhangi bir bellek bölgesinde aranmaz. Bu şekilde 6802 mikroişlemcisinde 25 komut vardır.

Örnek :

Assembly	op-code	yaptığı iş
ABA	1B	A = A + B
CLC	0C	C = 0

6. Akümülatör adresleme

Bu yöntem akümülatörün işlenen bilgiyi içerdiği, anlaşılır adreslemenin özel bir durumudur. 6802 mikroişlemcisinde 13 tane komut bu yöntemle adreslenebilmektedir. Bu adresleme doğrudan komuttan sonraki A ve B harfleri ile A akümülatörü veya B akümülatörü şeklinde tanımlanır.

Örnek :

Assembly	op-code	yaptığı iş
CLR A	4F	A acc. sil.
COM A	43	A = A’

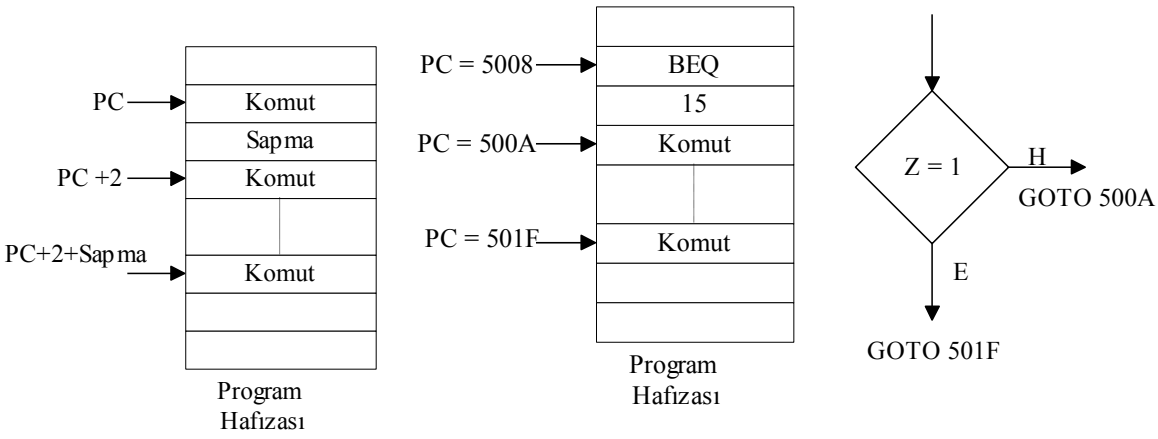
7. Relatif Adresleme

Sadece dallanma komutlarında kullanılan bu adresleme türünde, ulaşılması gereken adres program sayıcısının o andaki içeriğine bağlı olarak bulunur. Dallanma komutları iki byte den oluşur. Birinci byte işlem bytedir. İkinci byte ise öteleme (salınım, sapma, offset, dallanma) byte olup, program sayacına eklenir.

Bir dallanma sırasında ileriye gidilebileceği gibi, geriye de gidilmesi gerekebilir. Dolayısıyla öteleme sayısı işaretli bir sayıdır. Öteleme sayısı 8 bitlik bir sayı olup, bununla (00...FF) 256 durum ifade edilir. İşaret dikkate alınınca +127 ileriye (00...7F), -128 geriye (80...FF) gidilebilir. Fakat öteleme sayısı okunduğunda PC dallanma komutunun olduğu yerden 2 ilerisini göstereceği düşünülecek olursa, +129 ileri –126 geri dallanılabilir.

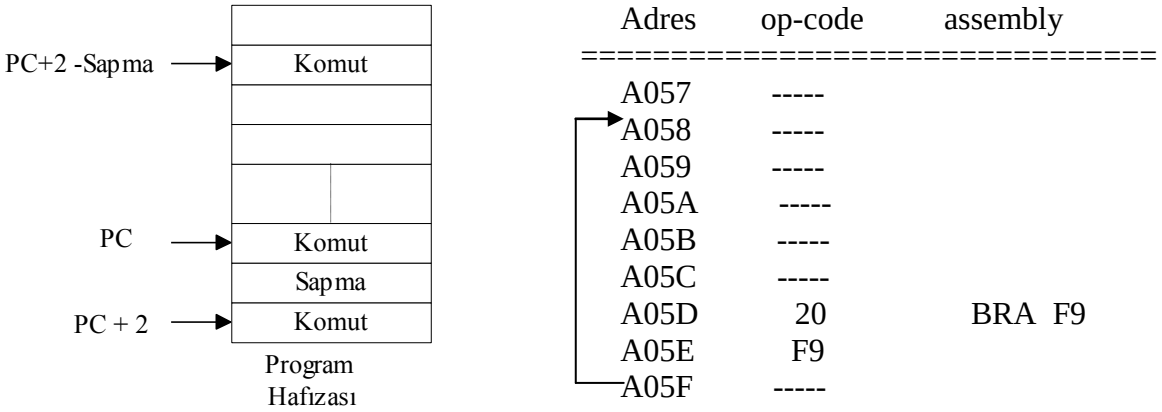
(PC + 2) –128 ≤ ÖTELEME MİKTARI ≤ (PC + 2) +127

- **İleri doğru sapma**



- **Geriye doğru sapma**

İleri doğru sapma yöntemiyle aynı prensiplere sahip bu yöntemde tek fark bağıl adresin negatif bir sayı olarak girilmesidir.



Geriye doğru sapma miktarını şu şekilde de bulunabilir :

Sapma komutundan itibaren FF den başlayarak geriye doğru sayılır, FF, FE, FD, FC,.....F9... sapılacak yere gelince durulur.

Problem: Aşağıdaki programda DNG1 ve DNG2 sapma miktarlarını 16 lık sayı sistemi tabanında veriniz.

```
        CLR A
DNG1    INC A
        NOP
        CMP A  #$10
        BNE  DNG1
        BRA  DNG2
        DEC A
DNG2    NOP
        WAI
```

5.5. MC6802 Assembly Uygulamaları

A) BİR ASSEMBLY PROGRAMIN YAZILIMDA İZLENECEK YOL

1. Probleme ait giriş/çıkış verileri ile istenen sonuç günlük dilde açık bir şekilde yazılmalı ve akış şeması çıkarılmalı
2. Kullanılacak olan bilgisayarın (mikroişlemcinin) kapasite ve özelliklerinin probleme cevap verip veremeyeceğinin belirlenmesi ve buna uygun mikroişlemci seçiminin yapılması.
3. Programın algoritmada belirtilen kurallara (adresleme modlarına) uygun olarak kodlanması.
4. Arzu edilen sonuçların elde edilip edilmediğinin kontrol edilmesi.

B) ÖRNEK PROGRAMLAR

1.

Komut	Adresleme Modu	Operasyonel Kodu	Yaptığı İş
ADD A #$33	Anında	8B 33	A = A + 33
ADD A $33	Doğrudan	9B 33	A = A + [33]
ADD A $0133	Genişletilmiş	BB 01 33	A = A + [0133]
ADD A $06,X	İndisli	AB 06	A = A + [X + 06]
ABA	Anlaşılır	1B	A = A + B
LDA B #$FF	Anında	C6 FF	B = FF
LDA B $55	Doğrudan	D6 55	B = [55]
STA A $1235	Genişletilmiş	B7 12 35	[1235] = A
INC A	Akümülatör (Anlaşılır)	4C	A = A + 1
LDX #$0011	Anında	CE 00 11	X = 0011
LDX $50	Doğrudan	DE 50	X = [0050, 0051]
JMP $0180	Genişletilmiş	7E 01 80	[0180] adresine git
JMP $02,X	İndisli	6E 02	[X + 02] adresine git

DEX	Anlaşılır	09	X = X - 1
CMP A #$50	Anında	81 50	A ile 50 i karşılaştırır
CMP $50	Doğrudan	91 50	A ile [0050] i karşılaş.
CMP A $0250	Genişletilmiş	B1 02 50	A ile [0250] i karşılaş.
ADC A #$05	Anında	89 05	A = A + C + 05
AND A #$1C	Anında	84 1C	A = A x 1C
ASL A	Akümülatör (Anlaşılır)	48	A sola kaydırılır
ASL $0D80	Genişletilmiş	78 0D 80	[0D80] sola kaydırılır
ROR $004E	Genişletilmiş	76 00 4E	[004E] sağa döndürülür
NOP	Anlaşılır	01	2 cp zaman geçiktirir
BCC xx	Relatif	24	C = 0 ise xx kadar sapar

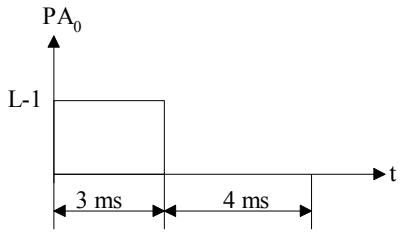
2. 55 adresinde CA sayısı ve B akümülatöründe 13 sayısı vardır.

ADD B #$55	Anında	CB 55	B = B + 55 = 68
ADD B $55	Doğrudan	DB 55	B=B+[0055]=13+CA=DD

3. Aşağıdaki programın çalışması sonucunda A akümülatöründe hangi sayı vardır.

LDA A #$22	86 22	A = 22
STA A $01C3	B7 01C3	[01C3] = A = 22
LDX #$0123	CE 01 23	X = 0123
ADD A $A0,X	AB A0	A=A+[X+A0]=22+[0123+A0]=22+[01C3]=22+22 = 44

4. 8000_H adresine yerleştirilmiş bir PIA nın PA_0 ucundan aşağıdaki şekilde bir kare dalga alınmak isteniyor. Gerekli assembly programı yazınız.




```
zz  LDA A  #$01
    STA A  $8000
    LDX  #$0000
xx  INX
    NOP
    CPX  #$00E7
    BNE  xx
    CLR A
    STA A  $8000
    LDX  #$0000
yy  INX
    NOP
    CPX  #$0134
    BNE  yy
    BRA  zz
```

Ödev : Yandaki programın operasyonel kodlarını çıkarınız

xx etiketindeki 3 ms'lik döngünün değeri, bu döngü içerisinde bulunan komutlarının çalışma sürelerinden çıkarılır :

INX → 4 + NOP → 2 + CPX → 3 + BNE → 4 = 13 cp; CPU → 1 Mhz ise 13cp = 13 µs

xx döngü miktarı = 3ms/13 µs = $(231)_{10}$ =$(00E7)_{16}$

Benzer şekilde 4ms lik yy döngüsünün değeri :

yy döngüsü miktarı = 4 ms / 13 µs = $(308)_{10}$ = $(0134)_{16}$

5. Kopyalama programı : 1000_H...2000_H arasındaki bilgileri 3000_H....4000_H arasına kopyalayan bir assembly program yazınız.

	LDX #$1000	CE 10 00	X = 1000
	STX $80	DF 80	[80] = X = 1000
	LDX #$3000	CE 30 00	X = 3000
	STX $82	DF 82	[82] = X = 3000
xx	LDX $80	DE 80	X = [80]
	LDA A $00,X	A6 00	A = [X+00]
	INX	08	X = X + 1
	STX $80	DF 80	[80] = X
	LDX $82	DE 82	X = [82]
	STA A $00,X	A7 00	[00 + X] = A
	INX	08	X = X + 1
	STX $82	DF 82	[82] = X
	CPX #$4001	8C 40 01	X – 4001 durumu CCR ye setler.
	BNE xx	26 ED	Z = 0 ise xx e dallanır
	SWI	3F	Dur

6. Aşağıdaki programın çalışması sonucu durum kod kayıtçısındaki H ve V bitlerinin durumu nedir?

LDA A #$E0	86 E0	A = E0	CCR = 1 1 H I N Z V C
LDA B #$09	C6 09	B = 09	CCR = 1 1 1 0 1 0 0 1
ABA	1B	A = A+B = E9	
TAP	06	CCR = E9	H = 1
SWI	3F		V = 0

7. Bilindiği üzere SWI kesmesi IRQ, NMI kesmelerinin aksine bir yazılım kesmesidir ve birçok mikrobilgisayarda mikroişlemciyi durdurucu bir işlev yapmaktadır. Bir SWI yazılım kesmesi programı yazınız.

xx	NOP	01	Sonsuz döngü
	BRA xx	20 FD	

8. Aşağıdaki programın çalışması sonucu indis kayıtçısında hangi sayı vardır?

LDS #$007F	8E 00 7F	SP = 007F
LDA A #$23	86 23	A = 23
LDA B #$34	C6 34	B = 34
PSH A	36	[007F] = 23
PSH B	37	[007E] = 34
LDX $7E	DE 7E	X = [007E, 007F] =3423
SWI	3F	

9. İki tane iki byte lık sayıyı toplamak için assembly program yazınız. Sayılardan biri 80_H ve 81_H adreslerine, diğeri ise 60_H ve 61_H adreslerinde saklıdır. Sonucu indis kayıtçısına yükleyiniz.

LDA A $81	96 81	A = [81]
LDA B $61	D6 61	B = [61]
ABA	1B	A = A + B
STA A $91	97 91	[91] = A
LDA A $60	96 60	A = [60]
ADC A $80	99 80	A = A + C + [80]
STA A $90	97 90	[90] = A
LDX $90	DE 90	X = [0090,0091]
SWI	3F	

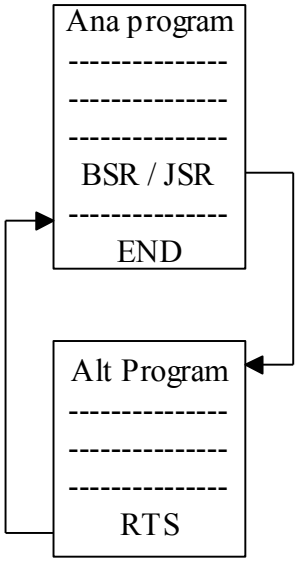
10. Aşağıdaki programın her komutunun çalışmasından sonra A birikecinde bulunan değeri yazınız.

LDA A #$23	A = 23
AND A #$F0	A = 20
COM A	A = DF
NEG A	A = 21
DEC A	A = 20
INC A	A = 21
ORA A #$34	A = 35
ASL A	A = 6A
LSR A	A = 35
SUB A #$43	A = F2
SWI	

Ödev : Aşağıda operasyonel kodları verilen programın assembly dil karşılığını yazarak ne iş yaptığını bulunuz?

CE
00
00
DF
50
4F
5F
9B
51
D9
50
08
DF
50
8C
01
2C
26
F4
97
53
D7
52
3F

C) ALT PROGRAM



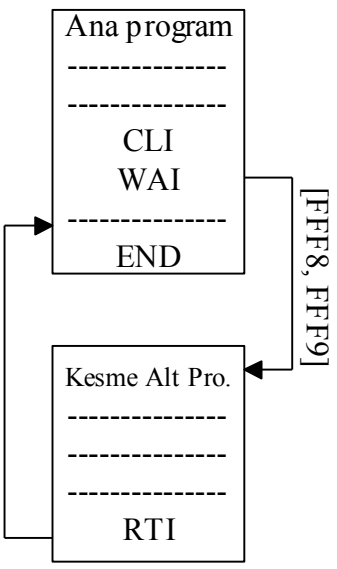
1. Alt programa giderken geri dönülecek adres değeri yığına atılır.
2. Alt program kullanılacaksa yığın göstergecinin (SP) başlangıç adresini ana programın başında mutlaka vermeliyiz ve verilen bu adresin RAM üzerinde olmasına dikkat edilmelidir.

Örnek : 0300_H ....0600_H adresleri arasına 02 yazan ve bir alt programda ise bunları toplayarak sonucu 0800_H adresine kayıt eden bir assembly programı operasyonel kodlarıyla birlikte veriniz.

```
                LDS  #$007F          8E  00 7F
                LDX  #$0300          CE  03 00
                LDA A  #$02          86  02
DONGU1          STA A  $00,X         A7  00
                INX                  08
                CPX  #$0601          8C  06 01
                BNE  DONGU1          26  F8
                BSR  DONGU2          8D  01
                SWI                  3F

DONGU2          CLR A                4F
                CLR B                5F
                LDX  #$0300          CE  03 00
DONGU3          ADD A  $00,X         AB  00
                ADC B  #$00          C9  00
                INX                  08
                CPX  #$0601          8C  06 01
                BNE  DONGU3          26  F6
                STA A  $0801         B7  08 01
                STA B  $0800         F7  08 00
                RTS                  39
```

D) KESME ALT PROGRAMI



1. Kesme alt programına giderken geri dönülecek adres değeri yığına atılır.
2. Kesme alt programı kullanılacaksa yığın göstergecinin (SP) başlangıç adresini ana programın başında mutlaka vermeliyiz ve verilen bu adresin RAM üzerinde olmasına dikkat edilmelidir.
3. IRQ kesme alt programı kullanılacak ise FFF8, FFF9 → adreslerine IRQ kesme alt programının başlangıç adresi önceden yazılmalıdır.
4. NMI kesme alt programı kullanılacak ise FFFC, FFFD → adreslerine NMI kesme alt programının başlangıç adresi önceden yazılmalıdır.

5. IRQ kesmesi kullanılacak ise, CCR deki I biti 0 yapılmalıdır. Bir IRQ kesmesi geldikten sonra, ikinci bir IRQ kesmesine izin vermemek için I biti 1 yapılmalıdır.

Örnek : Bir önceki alt program örneğini IRQ kesme alt programı ile yapınız.

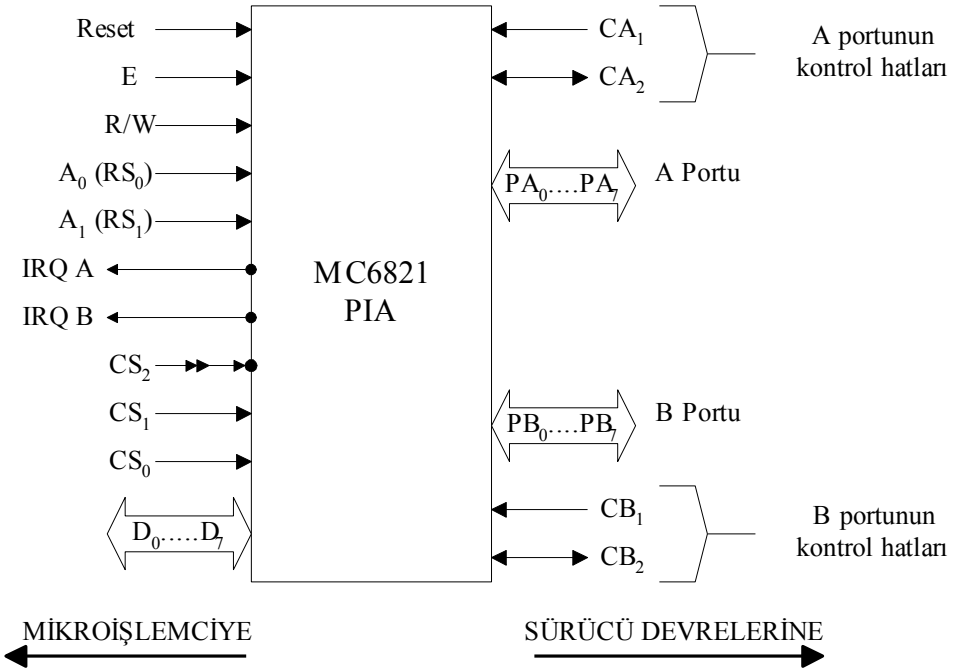
```
                LDS  #$007F          8E  00 7F
                LDX  #$0300          CE  03 00
                LDA A  #$02          86  02
DONGU1          STA A  $00,X         A7  00
                INX                  08
                CPX  #$0601          8C  06 01
                BNE  DONGU1          26  F8
                CLI                  0E
                WAI                  3E
                SWI                  3F

[FFF8, FFF9]    CLR A                4F
                CLR B                5F
                LDX  #$0300          CE  03 00
DONGU2          ADD A  $00,X         AB  00
                ADC B  #$00          C9  00
                INX                  08
                CPX  #$0601          8C  06 01
                BNE  DONGU2          26  F6
                STA A  $0801         B7  08 01
                STA B  $0800         F7  08 00
                RTI                  3B
```

5.6. Mikrobilgisayarlarda Giriş/Çıkış İşlemi

A) PIA (Peripheral Interface Adapter)

Paralel giriş / çıkış portudur. 68 serisi PIA: MC6821. PIA nın işlev tarzı (G/Ç portlarının ne kadarı giriş, ne kadar biti çıkış olduğu) assembly programın başlangıcında programlanmalıdır. Yani PIA nın her bir veri yolu giriş veya çıkış olarak programlanabilmektedir.



MC6821 de 3 tane 8 bitlik kayıtçı tipi vardır ;

1-) Veri kayıtçısı (Data Register → DRA, DRB) : Bu kayıtçı giriş yada çıkışa ait veriyi saklar.

2-) Veri yönü kayıtçısı (Data Direction Register → DDRA, DDRB) : Bu kayıtçıdaki her bir bit, buna karşılık gelen veri kayıtçısındaki bitin giriş yada çıkış olduğunu belirler. DDRA, DDRB de 0 → giriş, 1 → çıkış olarak tanımlıdır.

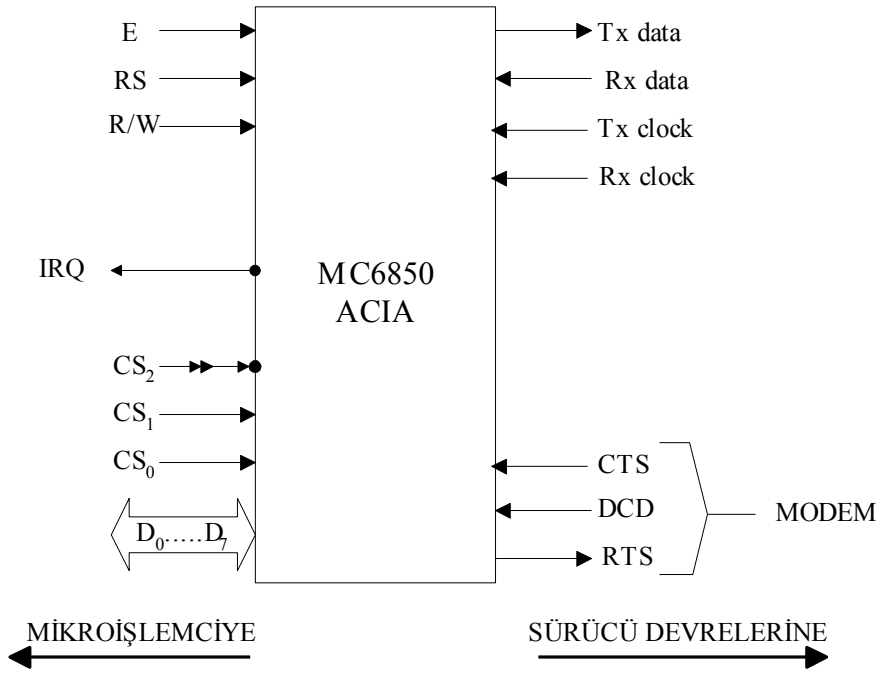
3-) Kontrol kayıtçısı (Control Register → CRA, CRB) : El sıkışma (handshake) için gerekli durum sinyallerini ve mantık bağlantılarını seçen bitleri üzerinde taşımaktadır

PIA CS sinyalleri ile seçilmektedir. Seçilme yapıldıktan sonra altı kayıtçıdan (DRA-DRB, DDRA-DDRB, CRA-CRB) her birine ulaşılabilir. Altı kayıtçı olmasına rağmen, bunları seçen iki adres yolu (RS_0-A_0, RS_1-A_1) vardır.

B) ACIA (Asychronous Communication Interface Adapter)

Asenkron olarak seri bilgi iletimi gerçekleştirir. 68 serisi ACIA MC6850 entegresidir. ACIA, bir mikroişlemcinin veri hattından aldığı paralel bilgiyi yazıcı, modem, gibi çevre birimlere seri biçimde olarak aktarabilen veya bu birimlerden aldığı seri biçimdeki bilgiyi mikroişlemcinin veri hatlarına paralel olarak verebilen eleman olup, UART (Universal Asynchronous Receiver

Transmitter – Evrensel Asenkron Alıcı Verici) olarak ta bilinir. Veri formatlama ve asenkron seri araçların bağlanabilmesi için kontrol sinyallerine sahiptir. Genel özelliklere ise aşağıdaki şekilde gösterilmiştir.



ACIA veri anayoluna ait paralel veriyi uygun format ve hata kontrolü ile seri olarak gönderebilmekte ve alabilmektedir. Dört tane kayıtçıya sahiptir. İki kayıtçı sadece okunabilir : Receiver (alıcı) ve Status (durum). Diğer ikisi ise sadece yazılabilir : Transmitter (gönderici) ve Kontrol kayıtçıları. Bu kayıtçılara RS ve R/W' sinyallerinin kombinasyonu ile ulaşılabilmektedir.

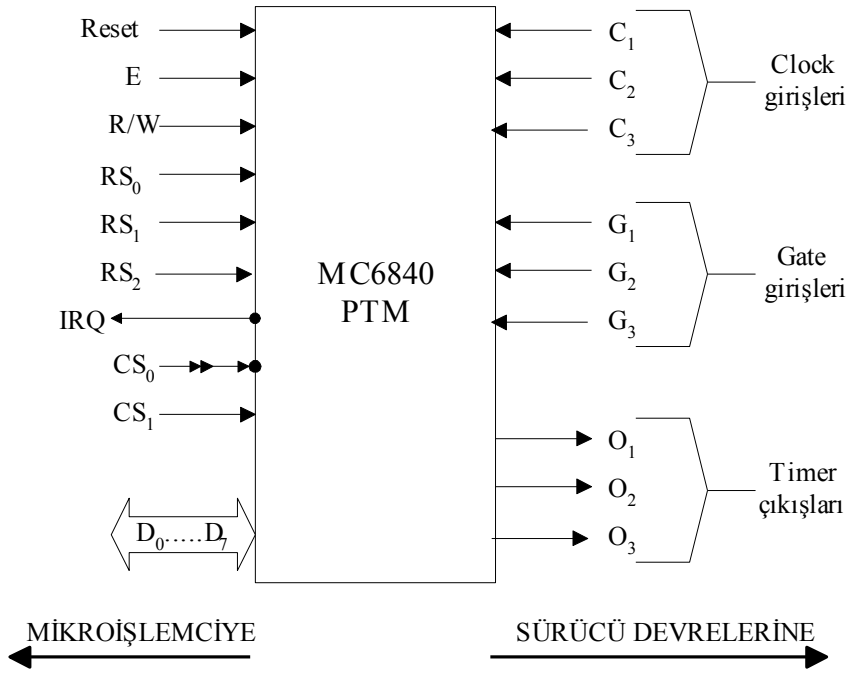
C) SSDA (Synchoronous Serial Data Adapter – Senkron Seri Veri Adaptörü)

Senkron olarak seri bilgi iletimi için kullanılır. 68 serisi olarak MC6852 mevcuttur. Bu birim birçok bakımdan ACIA ya benzer.

D) PTM (Programmable Timer Module – Programlanabilen Zamanlayıcı Modülü)

Mikroişlemcilerin endüstriyel uygulamalarında çok sık karşılaşılan bir gereksinim belirli olayları başlatma, bitirme ve uygun aralıklarla tekrarlamak için süre ölçmektir. Bir başka gereksinim de belirli olayları takip edip saymak ve belirli sayılarda belirli bazı işlemleri yapmaktır. Her iki işlem içinde mikroişlemci kullanılabilir. Örneğin süre ölçmek için mikroişlemci, başlama işareti ile birlikte bir döngüye sokulur ve durma işaretinde döngüden çıkılır. Döngü kaç defa yürütülmüşse, bir döngüyü yürütmek için gerekli süre o döngüdeki komutların yürütme sürelerinden bulunulabileceğine göre, toplam süre de bulunabilir. Aynı şekilde olay saymak için mikroişlemci bir bekleme durumuna sokulabilir, her bir olayda üretilen bir kesme, belirli bir

saklayıcının içeriğinin bir artırılmasını sağlayabilir. Fakat bu yöntemlerin sakıncası, mikroişlemcinin bu işlemleri yapabilmesi için zamanının büyük bir kısmını ayırması ve hatta bazı durumlarda başka hiçbir iş yapmamasıdır. Dolayısıyla süre ölçme ve olay sayma için genellikle ayrı bir donanım kullanılır. 68 serisinde bu amaçla kullanılabilecek MC6840 zamanlayıcı (timer) entegresi mevcuttur. Aşağıdaki şekilde MC6840 verilmiştir.



MC6840 ta 3 tane 16 bit ikili sistemde sayaç ve bunları kontrol eden üç kayıtçısı ile bir tane durum (status) kayıtçısı vardır. Kayıtçı seçici yolları RS_0, RS_1, RS_2 ve R/W yolları ile kayıtçılara, sayaçlara ve tutuculara (latch) ulaşabilmektedir. Zamanlayıcının (timer) işleyiş şekli kontrol kayıtçısına yazılan veri ile yönlendirilmektedir. 6840 değişik uygulama alanlarında kullanılacak şekilde tasarımlandırılmıştır.

BÖLÜM 6. HARVARD VE RISC MİMARİLİ MİKROİŞLEMCİ

6.1. PIC MİKRODENETLEYİCİLERİN TANITIMI

Mikrodenetleyicilerin kullanımı yaygınlaştıkça Atmel, Philips, Renesas, NEC, Microchip gibi firmalar mikrodenetleyicilerle piyasa çıkmaya başladılar. Bu firmalardan Microchip, 1990 yılından itibaren 8-bit'lik mimari üzerine yaptığı özel donanım eklentileri ile günümüzde onlarca çeşit mikrodenetleyici üretmektedir. Bu firma aynı zamanda 2004 yılı içerisinde dsPIC adı verdiği 16-bit mimarili yeni mikrodenetleyicisini çıkarmıştır. 8 bitlik mikrodenetleyiciler 8-bitlik veri yolu, 16-bitlik mikrodenetleyiciler ise 16-bitlik veri yolunu kullanırlar.

Microchip gibi bazı frmalar diğerlerinden farklı olarak uygulamalar için gerekli olabilecek çeşitli donanımları (ADC, DAC, RTC v.b.) mikrodenetleyici içerisine eklemektedir. Böylece bu donanımları harici olarak kullanmanın getireceği ek maliyet azaltılabilir. PIC mikrodenetleyicilerinin sağladığı avantajlar ;

- Piyasada kolay bulunabilmeleri ve birçok çeşidinin olması.
- Programlama için gerekli donanımların çok basit olması ve ücretsiz devre şemalarının kolaylıkla bulunabilmesi
- Programlama için gerekli olan yazılım geliştirme araçlarının Microchip tarafından ücretsiz olarak sunulması
- Sahip olduğu RISC mimarisinin, az sayıda komut ile kolayca programlanmasına olanak sağlaması
- Basic, C gibi yüksek ve orta seviyeli dillerde programlanmalarını sağlayan ücretli/ücretsiz yazılımlarının bulunması.
- Yaygın kullanımın bir sonucu olarak çok miktarda örnek uygulama ve kaynağın bulunması
- Microchip tarafından yazılan uygulama notlarının uygulama geliştirmede kolaylıklar sağlaması
- DIP kılıf yapısı ile de üretilmesinin kart tasarımında kolaylık sağlaması.

Bu avantajları ile PIC mikrodenetleyicileri, giriş seviyesindeki kullanıcılar için uygun bir başlangıç noktasıdır. Birçok karmaşık uygulama için bile farklı modeldeki PIC'ler ile çözümler üretebilmektedir.

A) PIC Mimarisi

Microchip firması tarafından üretilen mikrodenetleyicilerde Harvard mimarisi (RISC yapısı) kullanılmaktadır. Bu nedenle PIC mikrodenetleyicilerinin program ve veri belleği birbirinden ayrıdır. RISC yapısı nedeniyle PIC'ler oldukça az komut (35 komut) ile programlanmaktadır. Microchip, PIC mikrodenetleyicilerinin sınıfındaki diğer 8-bitlik mikrodenetleyicilere göre aynı işi

yapacak program kodunun 2 kat daha az yer kapladığını ve bu program kodunun 4 kat daha hızlı çalıştırdığını ileri sürmektedir.

PIC mikrodenetleyicilerinin program veri yolunun uzunluğu ise değişkendir. PIC mikrodenetleyicileri dış dünya ile haberleşirken 8-bit'lik veri yolu kullanılır. Microchip firması mikrodenetleyicilerini ailelere ayırırken "kelime uzunluğu" kriterini kullanmaktadır.

PIC aileleri de kendi aralarında kullanılan bellek yapısı, çalışma frekansı, giriş/çıkış uç sayısı ve özel amaçlı donanım gibi özellikleri ile birbirlerinden ayrılırlar. Bu teknolojik farklılıklardan öncelikli olarak bilinmesi gereken bellek yapısıdır.

B) PIC'lerin Diğer Mikrodenetleyicilere Göre Üstün Kılan Özellikleri

- **Kod Verimliliği :** PIC, Harvard mimarisi temelli 8 bit'lik bir mikrodenetleyicidir. Bu, program ve veri için ayrı ayrı BUS 'ların bulunduğu anlamına gelir. Böylelikle işleyiş miktarı veriye ve program belleğine eşzamanlı erişim sayesinde arttırılmış olur. Geleneksel mikrodenetleyicilerde veri ve programı taşıyan bir tane BUS bulunur. Bu, PIC 'le karşılaştırıldığın da işlem hızını en az iki kat yavaş olması demektir.
- **Güvenilirlik :** Tüm komutlar 12 veya 14 bitlik bir program bellek sözcüğüne sığar. Yazılımın, programın VERİ kısmına atlamaya ve VERİ 'yi komut gibi çalıştırmasına gerek yoktur. Bu 8 bitlik BUS kullanan ve Harvard mimarisi temelli olmayan mikrodenetleyicilerde gerçekleşmektedir.
- **Komut Seti :** 16C5x ailesi yazılım oluşturmak için 33 komuta sahip iken, l6Cxx parçaları içinse bu sayı 35 dir. PIC tarafından kullanılan komutların hepsi kayıtçı (register) temellidir ve 16C5x ailesinde 12 bit, 16Cxx ailesindeyse 14 bit uzunluğundadır. CALL, GOTO ve bit test eden BTFSS, INCFSZ gibi komutlar dışında her bir komut, tek bir çevrimde çalışır.
- **Hız :** PIC, osilatör ve yerleşik saat yolu (clock bus) arasına bağlı yerleşik bir) 4'lü bölünmeye sahiptir. Bu, özellikle 4 MHz 'lik kristal kullanıldığında komut sürelerinin hesaplanmasında kolaylık sağlar. Her bir komut döngüsü 1 μs dir. PIC oldukça hızlı bir mikroişlemcidir. Örneğin 5 milyon komutluk bir programın, 20 MHz 'lik bir kristalle örneklenmesi yalnız 1 sn sürer. Bu süre Intel 386SX 33MHz 'in hızının neredeyse 2 katıdır.
- **Statik işlem :** PIC tamamıyla statik bir mikroişlemcidir. Başka deyişle saati durdurduğunuzda, tüm kayıtçı içeriği korunur. Pratikte bunu tam olarak gerçekleştirmek mümkün değildir. PIC uyuma (standby) moduna alındığında, saat durur ve PIC 'i uyutma işleminden önce hangi durumda olduğunu kullanıcıya hatırlatmak için çeşitli bayraklar kurulur. PIC uyuma modunda yalnızca 1 μA den küçük bir değere sahip bekleme (standby) akımı çeker.
- **Sürücü Kapasitesi :** PIC yüksek bir sürücü çıktı kapasitesine sahiptir.

- **Seçenekler :** PIC ailesinde her tür ihtiyacı karşılayacak çeşitli hız, sıcaklık, kılıf, I/0 hatları, zamanlama (timer) fonksiyonları, seri iletişim portları, A/D ve bellek kapasite seçenekleri bulunur.
- **Güvenlik :** PIC endüstride en üstünler arasında yer alan bir kod koruma özelliğine sahiptir. Koruma bitinin programlanmasından itibaren, program belleğinin içeriği, program kodunun yeniden yapılandırılmasına olanak verecek şekilde okunamaz.
- **Geliştirme:** PIC, geliştirme amacıyla yeniden programlanabilen (EPROM, EEPROM) ve seri üretim amacıyla OTP (one time programmable - bir kere programlanabilir) için pencereli yapıda bu1unabilir. Geliştirme araçlarının temini mümkündür ve fiyatları bir ev kullanıcısı için bile satın alınabilir düzeydedir.

C) PIC Donanım Özellikleri

PIC Çeşitleri : Microchip ürettiği mikrodenetleyicileri 4 farklı aileye ayırarak isimlendirmiştir. PIC ailelerine isim verilirken sözcük uzunluğu göz önüne alınmıştır. Bir CPU dahili veri yolu uzunluğuna sözcük uzunluğu denir. Microchip PIC 'leri 12/14/16 bitlik sözcük uzunluklarında üretilmektedir ve buna göre aşağıdaki aile isimlerini verilmektedir:

- PIC16C5XX ailesi 12-bit sözcük uzunluğu,
- PIC16CXXX ailesi 14-bit sözcük uzunluğu,
- PIC17CXXX ailesi 16-bit sözcük uzunluğu,
- PIC12CXXX ailesi 12-bit/14-bit sözcük uzunluğuna sahiptir.

Tablo 6.1. PIC 12,16,17 ve 18XXX serilerinin genel özellikleri

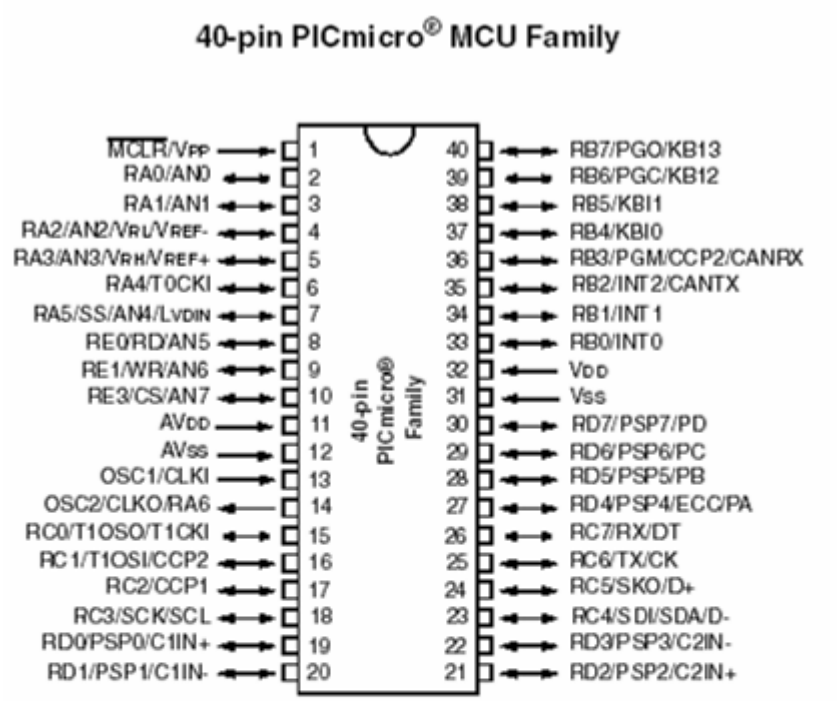
Seri Adı	Program Belleği	OTP/FLASH Belleği	EEPROM Belleği	RAM Belleği	ADC Kanalı	G/Ç Port	Seri Port	PWM Kanalı	Hız MHz
12XXX	3568	2048×12 bit	16	128	4(8bit)	6	-	-	10
16XXX	14336	8192×14 bit	256	368	10(12bit)	52	var	2	24
17XXX	32768	16384×16	-	902	16(10bit)	66	var	3	33
18XXX	32768	16384×16	-	1536	8(10bit)	34	var	2	40

Bir CPU, yonga dışındaki harici ünitelerle veri alışverişini kaç bitle yapıyorsa buna veri yolu (DATA BUS) bit sayısı denir. PIC 'ler farklı sözcük uzunluklarında üretilmelerine rağmen harici veri yolu tüm PIC ailesinde 8-bittir. Yani bir PIC, I/O portu aracılığı ile çevresel ünitelerle veri alışverişi yaparken 8-bitlik veri yolu kullanır.

PIC programlayıcıları, program kodlarını yazarken bir komutun kaç bitlik bir sözcük uzunluğundan oluştuğuyla pek fazla ilgilenmezler. Seçilen bir yongayı programlarken uyulması gereken kuralları ve o yongayla ilgili özelliklerin bilinmesi yeterlidir. Bu özellikler; PIC 'in bellek

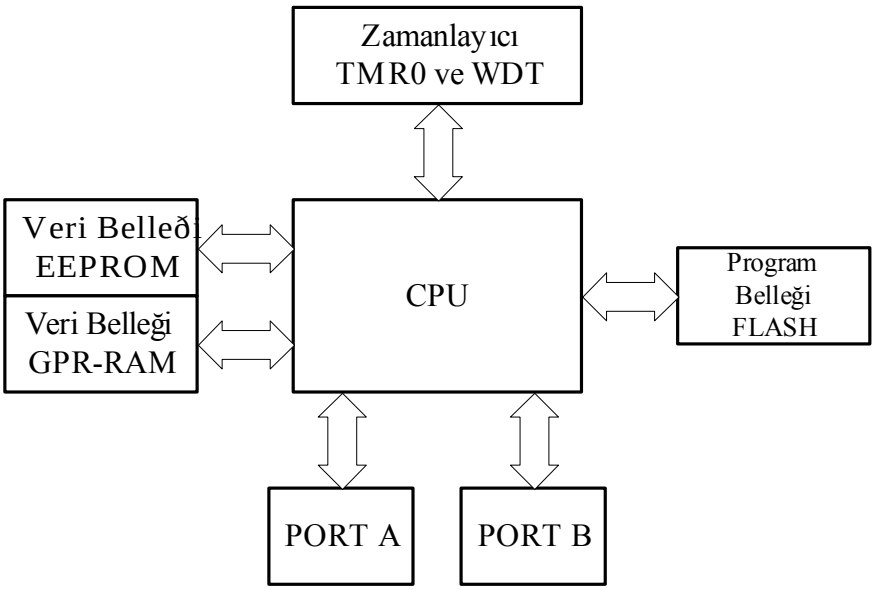
miktarı, I/O portu sayısı, A/D dönüştürücüye sahip olup olmadığı, kesme fonksiyonlarının bulunup bulunmadığı, bellek tipinin ne olduğu (Flash, EPROM, EEPROM vb.) gibi bilgilerdir.

PIC 'lerin Dış Yapıları

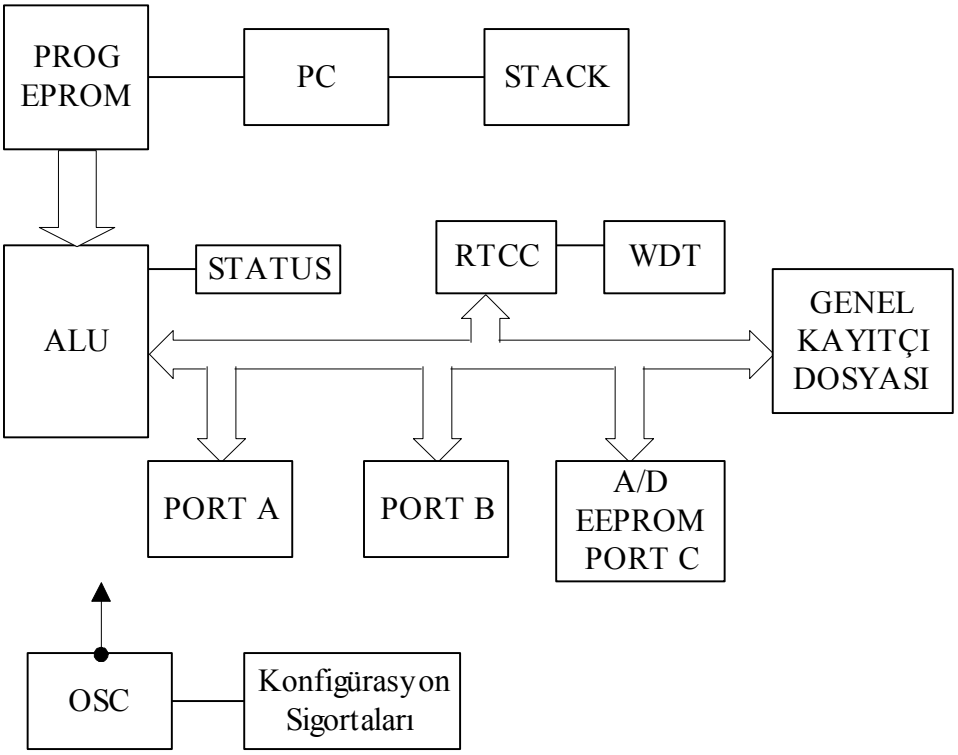


PIC16CR65	PIC16C765	PIC18F4220
PIC16C65B	PIC16C774	PIC18F4320
PIC16C67	PIC16F871	PIC18F4331
PIC16C662	PIC16F874	PIC18F4431
PIC16C74B	PIC16F874A	PIC18F442
PIC16C77	PIC16F877	PIC18F452
PIC16F74	PIC16F877A	PIC18F448
PIC16F77	PIC18C442	PIC18F458
	PIC18C452	

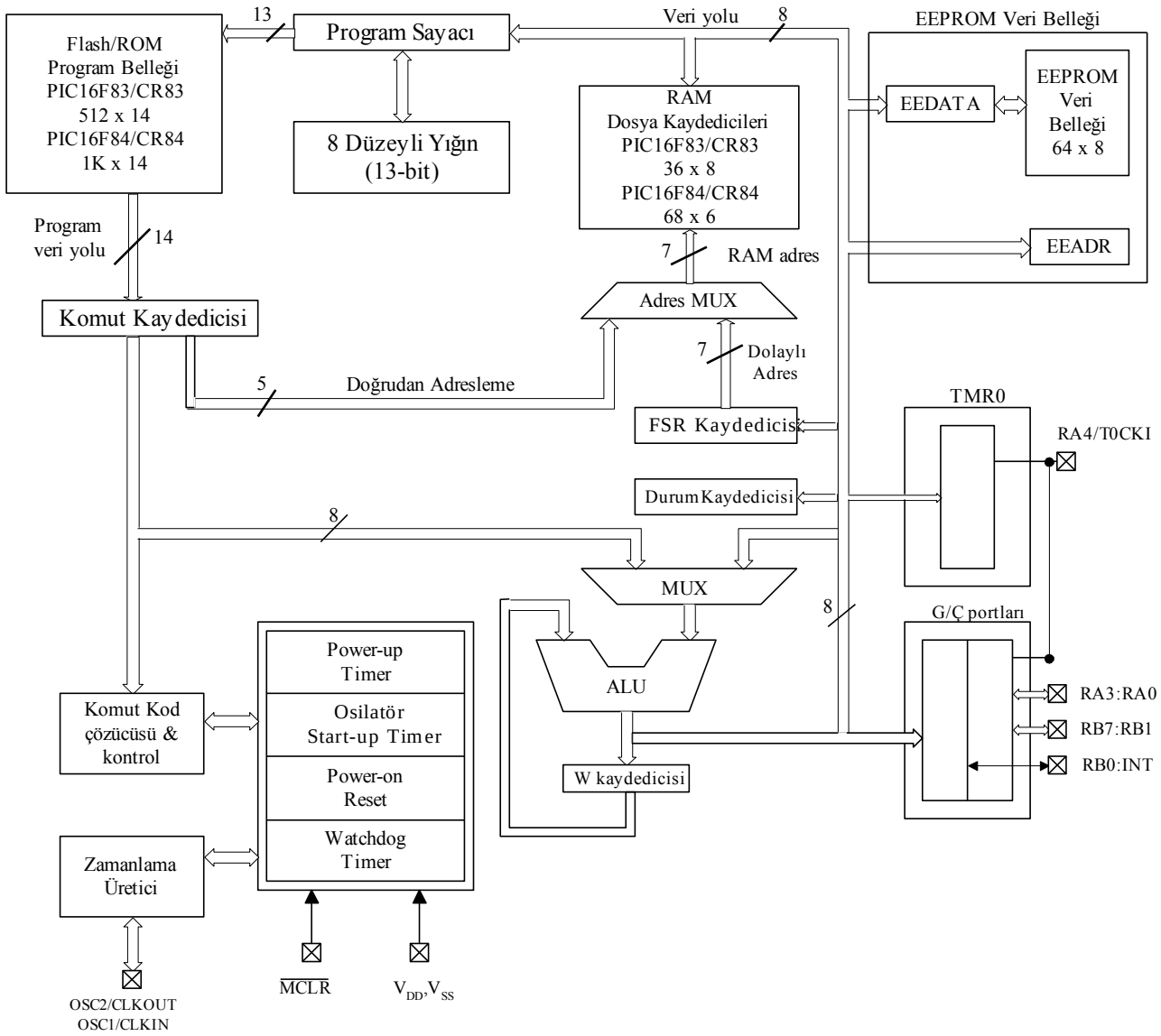
PIC İç Yapısı



Şekil 6.1. PIC 16F8X mikrodenetleyicilerinin genel blok diyagramı



Şekil 6.2. Temel PIC Blok Diyagramı



Şekil 6.3. PIC mikrodenetleyicilerinin genel blok diyagramı

D) PIC'lerin Temel Elemanları

➢ ***Aritmetik Mantık Birimi (Arithmetic Logic Unit - ALU) :*** CPU 'nun kalbi olup, adından da anlaşıldığı gibi komut sözcüğüne (Instruction Word) göre aritmetik ve mantık işlemlerini yapar. Komut sözcüğünün başlıca dört biçimi olup, komut sözcüğünü bu biçimlere göre ayrıştırıp, uygularken W kayıtçısını ve gerekiyorsa diğer kayıtçıları da kullanır. ALU içerisinde toplama (ADD), çıkartma (SUB), bir kayıtçısının sağ ve sol bitlerinin yerini değiştirme (SWAP), kaydırma (SHIFT) ve döndürme (ROTATE), ... gibi işlemleri yapan birimler vardır. Ayrıca AND, OR, XOR mantıksal işlemlerini gerçekleştiren birimler de bulunmaktadır. ALU, veri iletişim hattı aracılığıyla verileri alır, komuta göre işler ve ilgili birimleri uyararak sonucu, W ya veya komutta belirtilen hedef kayıtçısına yükler. Bu durum yine, mikrodenetleyiciye göre değişir. Çünkü, bazı mikrodenetleyiciler sonucu yalnız W, yani akümülatöre yazarken, bazıları hem akümülatöre hem de kayıtçıya yazabilir. Örnek olarak, PIC16F87X ailesinin, toplama, çıkarma ve benzeri işlemlerin sonucunu istenirse W da veya istenirse kayıtçıda tutma esnekliği vardır. PIC, diğer mikroişlemcilerden, aritmetik ve mantık işlemleri için bir tek ana kayıtçıya sahip oluşuyla farklıdır.

➢ ***Akümülatör/Working Register :*** Genel amaçlı bir kayıtçıdır. W kayıtçısı 8 bit genişliğindedir ve CPU daki herhangi bir veriyi transfer etmek üzere kullanılır. ACC / A / W olarak kısaltılır. Tüm aritmetik ve mantık işlemlerinde, işlenenlerin ve bazı mikroişlemcilerde de hem işlenen hem de işlem sonuçlarının tutulduğu bir kayıtçıdır. Verilerle ilgili kaydırma, döndürme, eksiltme, arttırma, karşılaştırma ve tersini alma işlemlerinin gerçekleştirilmesi ile bu işlemlerin sonuçlarının tutulmasında kullanılır. Akümülatörün bu özellikleri, mikroişlemciden mikroişlemciye değişebilir. Özellikle mikrodenetleyicilerde akümülatöre (W kayıtçısına) bazı ek işler yüklenebilir. Microchip firması, kendi ürünlerinde akümülatör yerine working register (W) ismini kullanmaktadır.

➢ ***Veri Kayıtçı Dosyaları (Data Register Files) :*** CPU alanında bulunur ve iki kategoriye ayırılır: I/O ve Kontrol şeklinde çalışanlar ve tamamen RAM gibi çalışanlar.

➢ ***BUS :*** Harvard Mimarisi temeli mikrodenetleyicilerde, veri akış miktarını hızlandırmak ve yazılım güvenliğini arttırmak amacıyla ayrı BUS 'lar kullanılır. Bu mimari, veri ve program belleğine eşzamanlı erişimi olanaklı kılar.

➢ ***Program Sayacı (Program Counter, PC) :*** Mikroişlemci (CPU) tarafından yürütülecek komutun, program belleğindeki adresini tutar. PC kayıtçısının içinde, bulunulan yeri gösteren adres olduğu için, kendisi bir göstergedir (Pointer). Program sayacında, ilk komut çalıştırıldıktan sonra, ikinci komutun bulunduğu adres oluşur. Böylece program sayacı, sürekli bir sonra çalıştırılacak komutun adresini gösterir.

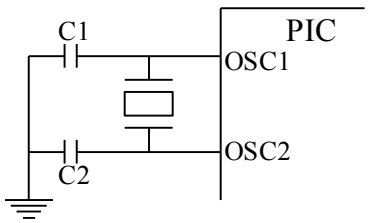
➢ ***Stack (Yığın) :*** PC, altprogram tamamlandığında, yani altprogramın bütün komutları çalıştırılıp bitince, altprogramın başlatılmasından hemen önceki adrese geri döner. Bunun için, altprogramın çalıştırılmasından bir önceki adres, yığın (stack) ismi verilen bir dizinin en üstüne konur (push). Bu işlemden sonra, PC altyordamın içindeki ilk komutun adresini alır ve altyordamın her komutunda, birer birer artmayı sürdürür. Altyordamdan dönüş komutu Return 'e geldiğinde, yığının en üstüne konan adres PC 'ye geri yüklenir. Böylece programda, altyordamın çağırıldığı noktaya geri dönülmüş olur. Bir altprogram içinden, başka bir altprogramı çağırdığımızda da yine aynı işlemler yapılarak, PC ve yığın aracılığıyla çağırıldığı altyordama geri dönebilir. Geri dönüşü sağlayan mekanizma, yine yığındır. Yığın, FILO (First In Last Out - İlk Giren Son Çıkar) mantığına göre işleyen bir kayıt alanıdır. Mikrodenetleyiciye göre değişen yığının boyutu, o mikrodenetleyicinin iç içe yürütebileceği, çağrılabilecek altprogramların sayısını belirler. Yığın veya yığının herhangi bir elemanına, programcı tarafından hiçbir yolla erişilemez, içeriği okunamaz veya üzerine yazılamaz. PIC16F8X ve 16F87X ailelerinde, yığın boyutu sekizdir. Bunlarda, iç içe en fazla sekiz altprogram kullanılabilir. Yığında kesme (interrupt) işlemleri de, altprogramlar gibi bir yer tutar. Programda

yığın taşması (stack overflow) hatasına düşmemek için, iç içe çağırılan altprogram ve kesme altprogramlarının sayısını, programcının denetlemesi gerekir. Yığının her elemanı 13 bit uzunluğundadır. PIC16F8X, 16F87X ailelerinde kayıtçılara yada veri, program belleğine, doğrudan veya dolaylı erişilebilir.

➢ ***Status (Processor Status Register - İşlemci Durum Yazmacı):*** Kısa adı PS veya STATUS olan bu yazmaç, sekiz bitliktir. İçinde çeşitli durumları bildiren uyarı bitleri bulunduğu için, *bayrak (flag)* kayıtçısıda denir. Mikroişlemcinin o andaki durumunu bildirir. Bu kayıtçıya bakılarak, yapılan işlemin sonucu hakkında bilgi alınabilir. Aritmetik işlemlerde; elde olup olmadığı, sonucun sıfır olup olmadığı, status kayıtçısının ilgili bitlerine bakılarak öğrenilir. Status kayıtçısında, dolaylı adresleme ve doğrudan adresleme bilgileri de bulunur. Program, status kayıtçısından öğrenilen bilgilere göre yönlendirilir. Üreticiler kendi mikrodenetleyicilerindeki status kayıtçılarında, başka özel durumlar içinde bitler ayırmışlardır. PIC16F87X serisi için bu yazmaç ilerde ayrıntılı olarak tanımlanacaktır.

PIC 'lerde bunların dışında, dolaylı erişim için INDF ve FSR; kesmeler için INTCON; zamanlama için TMRO, TMR1-2 ve girdi ile çıktılar için TRISA, TRISB,...,TRISE ile PORTA, PORTB,…,PORTE gibi pek çok kayıtçı vardır. Bunları, ilerdeki bölümlerde daha ayrıntılı olarak göreceğiz.

➢ ***OSC (Zamanlama ve Denetim Bölümü):*** Mikrodenetleyicinin kendisine verilen komutları işleyebilmesi için, saat (clock) denilen, kare dalga işareti gerekir. Bu işareti, mikrodenetleyici içerisinde bulunan bir osilatör devresine, dışarıdan bağ1anan bir kristal üretir. Üretilen işaret, komutların işlenmesinde zamanlamayı sağlar. Hassas zamanlamanın gerekli olmadığı uygulamalarda, RC osilatör kullanmak maliyeti azaltır. RC osilatörün rezonans frekansı besleme voltajına, çalışma sıcaklığına, R direncinin ve C kondansatörünün değerlerine bağlıdır. RC osilatör tasarımında direnç değeri 5-100 KΩ aralığında olmalıdır. 500 KΩ gibi yüksek R değerleri osilatörü gürültü, nem ve sızıntıya duyarlı duruma getirir. R değerleri 2.5 KΩ altında ise kararsızlığa hatta osilasyon kaybına yol açabilmektedir. Osilatör frekansı mikrodenetleyicide 4'e bölünür ve diğer devrelerle senkronizasyonu sağlamak için kullanılır.



Şekil 6.4. OSC'nin Bağlanması

➢ ***Program Belleği :*** Önce bellek kullanımında sıkça başvuracağımız bazı terimleri tanımlayalım. Bu terimleri, program belleği ve veri belleğinin anlatılacağı kesimlerde kullanacağız. Bunlar:

- *Bellek büyüklüğü (Memoy size):* Bellekte tutulabilen veri miktarıdır
- *Erişim süresi/Okuma çevrim süresi (Acces time):* Bellekten okuma işleminin başlangıcından, gerekli bilginin alınışına kadar geçen süredir.
- *Erişim çevrim süresi (Acces cycle time) :* Bellekten ard arda iki okuma arasındaki süredir.
- *Yazma çevrim süresi (Write cycle time) :* Belleğe ard arda iki yazma arasındaki süredir.

Program belleği mikrobilgisayar uygulaması için verilen komutlardan oluşan programın, yerleştiği alandır. Programın bulunduğu ilgili adresler program sayacında tutulur. P16F87X ailesinde üç bellek bloku bulunur. Bunlar program belleği, veri belleği (RAM) ve EEPROM veri belleğidir. Program ve veri belleğinin erişim yolları ayrıdır.Program belleği de kendi içinde sayfalara ayrılır. Program belleği 16F877 de 8 sözcük uzunluğudur. 16F87X ailesinde her bir sayfa iki sözcük uzunluğu büyüklüğündedir. Bellek sayfa sayısı arttığında, adresleme için yalnız PC kayıtçısının kullanılması yetmez. Buna ek olarak, bellek sayfalarına erişim sayısının da tutulması gerekir. 16F87X ailesinde, PCLATH kayıtçısı, doğru sayfadaki adrese erişmek için PC ile birlikte kullanılır. İlerde PC ve PCLATH birlikte daha ayrıntılı incelenecektir.

➢ ***Veri Belleği :*** Programın çalışması için, veri belleğindeki kayıtçılar kullanılır. Dosya kayıtçılarının uzunluğu 8 bittir. Yalnız, PCLATH kayıtçısı 5 bit uzunluğundadır. Dosya kayıtçıları özel veri bellek alanındadır. Yani bunların adresleri önceden belirlenmiştir. Bunların dışındaki veri alanları program içinde kullanılmak istenen geçici değişkenler için atanabilir.

➢ ***EEPROM:*** 16F877 nin veri alanları dörde ayrılır. Bunların her birine bank denir ve bank0 dan başlayarak bank3 'e kadar ulaşan veri belleği vardır. Her EEPROM veri belleği bloğu, 000_H dan $07F_H$ adresine kadardır (128 Byte). Aşağıdaki tabloda 16F877 nin veri belleği haritası gösterilmiştir. Kayıtçılar ilerideki bölümlerde daha ayrıntılı anlatılacaktır.

➢ ***Giriş/Çıkış Birimi :*** Giriş birimi mikrodenetleyici dışındaki devreler ve sistemlerden gelen işaretleri, mikrodenetleyiciye aktaran tümleşik bir devre (Integrated Circult - IC) dir. Benzer şekilde, çıkış birimi de yonganın çıkış işaretlerini, mikrodenetleyici dışındaki devrelere aktaran tümleşik bir devredir. Uygulamada iki IC, aynı yonga içinde üretilir. Bu nedenle, iki IC nin de denetlenmesi amacıyla, bir de kontrol devresi eklenmiştir. Mikrodenetleyicinin dış dünyayla iletişimi, giriş/çıkış (I/O) portları ile kurulur. Portların iletişim özellikleri, seçilen PIC 'e göre farklılıklar gösterir. P1C16F877 'de Giriş/Çıkış portları A dan E ye kadar harflerle belirtilir. İlerdeki bölüm giriş çıkış birimi ayrıntıları belirtilip; hangi pinlerle, ne tür giriş ve çıkış özellikleri sağlandığı tablolar yardımıyla gösterilecektir. Bu bacaklar yoluyla, uygulama için geliştirilen elektronik kartlara bağlı olarak, potansiyometre, röle, sensör, klavye, yazıcı, LCD, 7SD, ... gibi bir

birimi bağlayarak, PIC 'e girdi verilebilir veya PIC 'ten çıktı alınabilir. Bu elemanlar kullanılarak haberleşirken PIC 'in A/D çevirici, Paralel Slave Port, USART, MSSP, Capture/Compare/PWM, ... gibi modülleri kullanılabilir.

- ***RTCC (Real Time Clock Counter – Gerçek zamanlı sayaç) :*** RTCC için gerekli olan saat darbeleri bir dahili veya harici işaretle sağlanır. Harici işaretler, yanlış sayımların mümkün olduğunca önlenmesi için tamponlanır. Hız kontrolü, oran göstergeleri veya frekans sayaçları gibi uygulamalarda RTCC bir harici saat besleyiciyle kullanılır. Dahili saat besleyici, *multitasking (çok görevlilik)* uygulamalarını (veri iletişimi gibi) yürüten bir yazılım olan RTOS (Real Time Operating System - Gerçek Zamanlı İşletim Sistemi) nin bir bölümü olarak kullanılabilir. RTCC, girdi işaretini yazılımın daha rahat taşıyabileceği bir orana getirilmek üzere ortalamaya çeken veya yavaşlatan bir pre-scalarla ayarlanabilir.

- ***WDT (Watch-dog Timer - Zamanlayıcı) :*** WDT 'nin kullanım amacı, PIC 'i veya herhangi bir işlemciyi bir döngüde kilitlenmekten uzak tutmaktır. Böyle bir durum yazılımda bir hata veya harici bir elektriksel kıvılcımlar nedeniyle ortaya çıkabilir. WDT, PIC 'e bir çeşit kalp atışı sağlar ve eğer WDT yazmacını düzenli aralıklarla temizlenmezse bu kalp atışları PIC 'i resete zorlar. Bu mükemmel bir özelliktir ve Güvenlik Sistemleri gibi ana kontrol panelinde bir kilitlenmenin asla söz konusu olmaması gereken uygulamalar için birebirdir. Güç koruması gereken ve yalnızca periyodik olarak açılması gereken ürünlerde WDT 'den faydalanırlar.

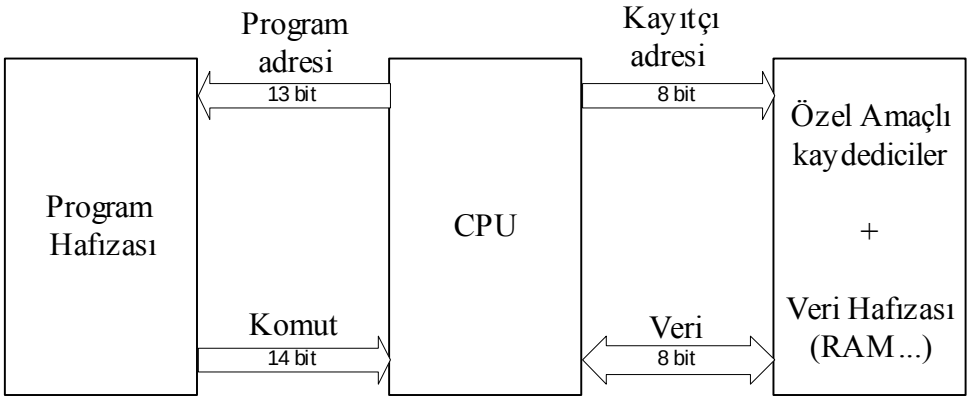
6.1. Bölüm Kaynakları

1. *O. Altınbaşak, 2001. "Mikrodenetleyiciler ve PIC Programlama", Atlaş Yayıncılık, İstanbul.*
2. *N. Gardner, 1998. "PIC Programlama El Kitabı", Bileşim Yayıncılık, İstanbul.*
3. *O. Urhan, M.Kemal Güllü, 2004. "Her Yönüyle PIC16F628", Birsen Yayınevi, İstanbul.*
4. *N. Topaloğlu, S. Görgünoğlu,2003. "Mikroişlemciler ve Mikrodenetleyiciler", Seçkin Yayıncılık, Ankara.*
5. *Y. Bodur, 2001. "Adım Adım PICmicro Programlama",İnfogate.*
6. *"PIC16F87x Data Sheet", Microchip Technology Inc., 2001.*

6.2. PIC16F877 MİKRODENETLEYİCİSİ

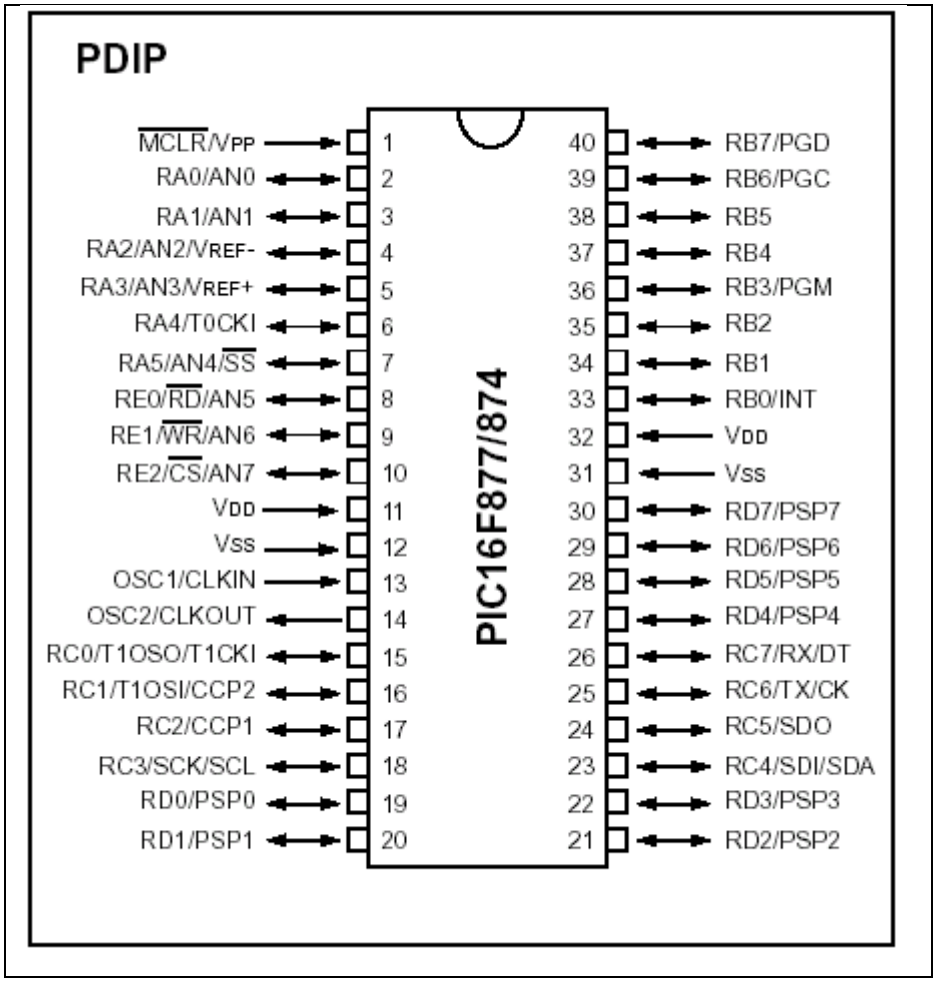
A) Genel Özellikleri

- PIC 16F87X serisi PIC 16CXX ailesinin özelliklerini taşır.
- PIC- I6CXX de Harvard mimarisi kullanılmıştır: veri yolu 8 bit genişliğinde, program belleğine program yolu yada adres yolu (program bus / address bus) denilen 13 bit genişliğindeki diğer bir yolla erişilir. PIC 16C87X de komut kodları (opcode), 14 bittir. 14 bitlik program belleğinin her bir adresi, bir komut koduna (Instruction code/word) karşılık gelir.



Şekil 6.5: PIC16F877 nin Harvard Mimarisi

- Her komuta bir çevrim süresinde (saykıl, cycle) erişilir ve komut yazmacına yüklenir. Dallanma komutları dışındaki bütün komutlar, aynı çevrim süresinde çalıştırılırlar. Bu sırada program sayacı, PC bir artar.
- Dallanma yada sapma komutları ise, iki ardışık periyotta çalıştırılır ve program sayacı PC, iki arttırılır.
- RISC mimarisine göre tasarlanmışlardır.
- Dış mimarisi : 40 pin



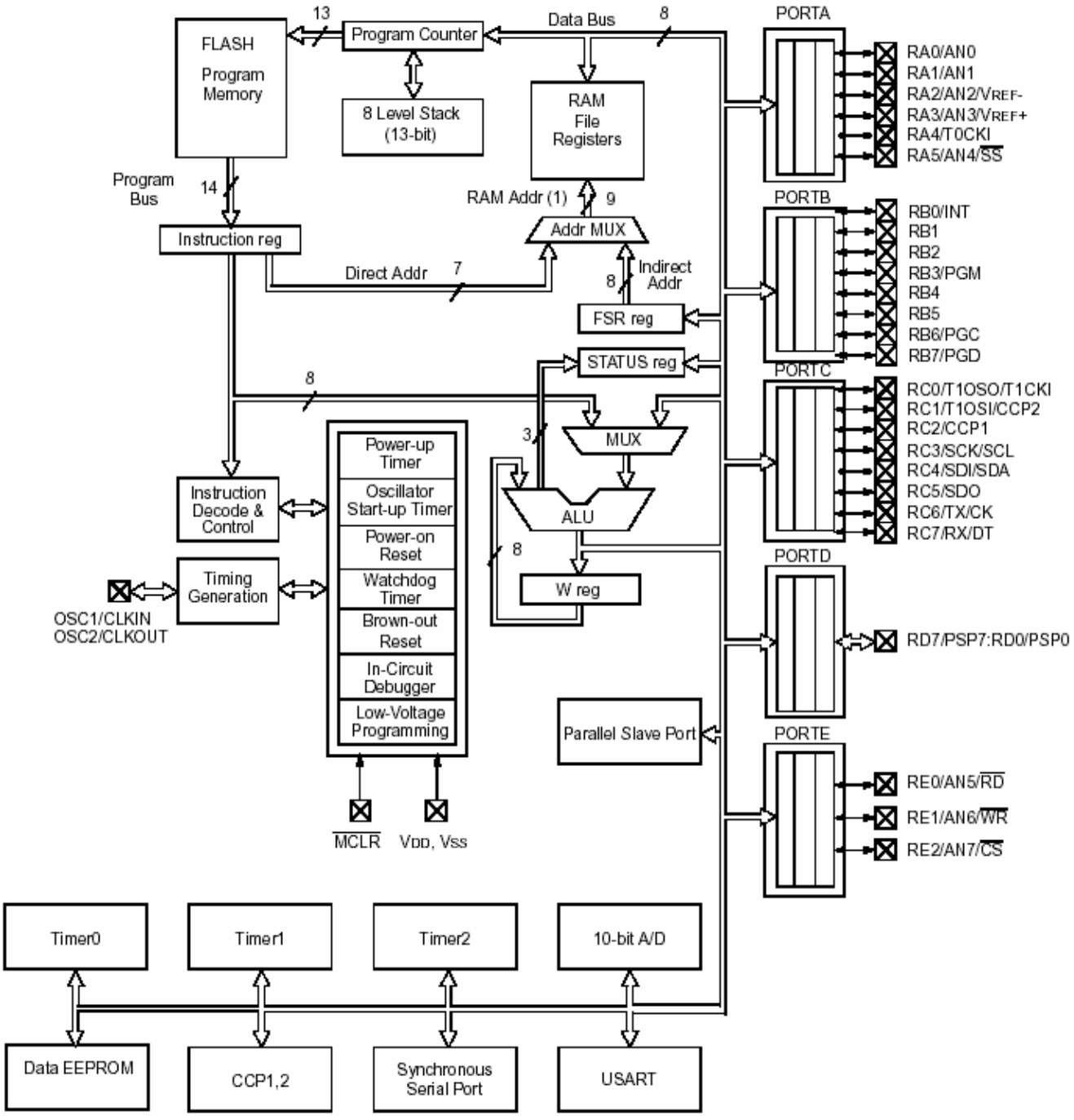
Şekil 6.6 : PIC16F87X Dış Mimarisi

- 35 tek sözcük uzunluğunda (bir sözcük uzunluğu =14-bit) komuta sahiptir.
- Clock hızı : 20 MHz olup, bir komut saykılı

  tcp = 1/ (20/4) = 0.2 μsn = 200 nsn

B) İç Mimarisi

Aşağıdaki şekilde PIC16F877 yongasının temel donanım yapısı görülmektedir.

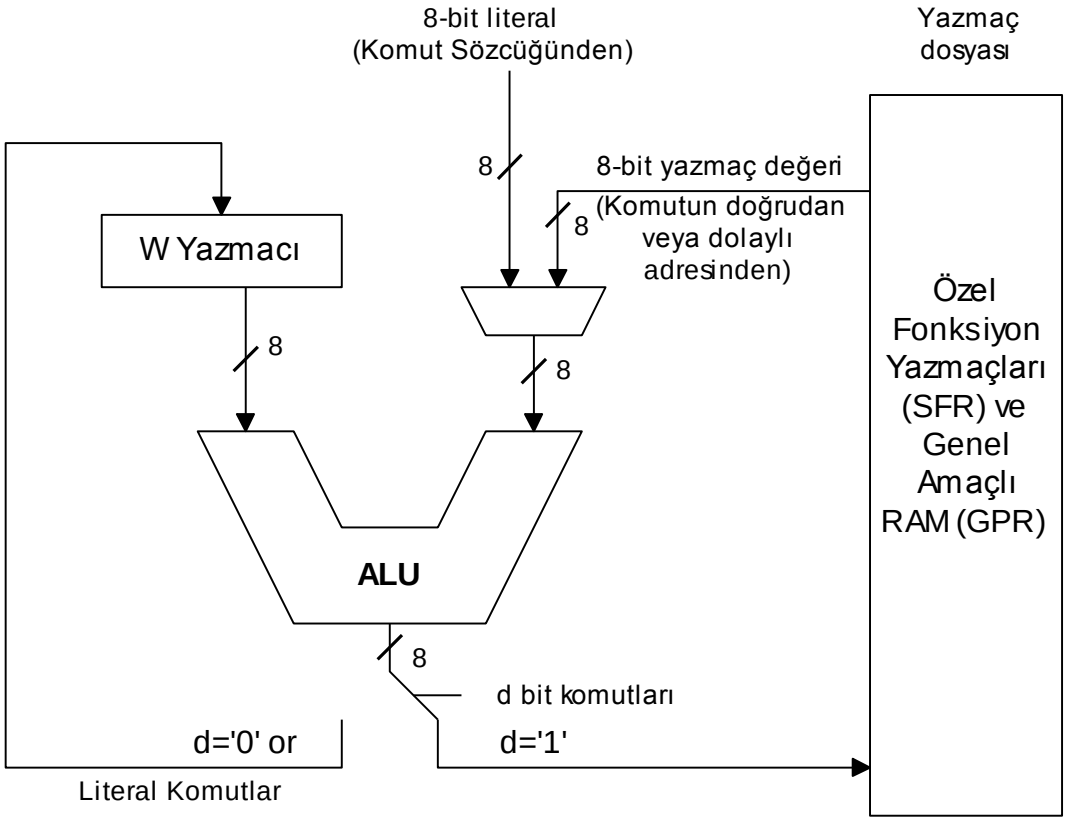


Şekil 6.7: PIC16F87X İç Mimarisinin Blok diyagramı

Blok şemadan;

1. Aritmetik ve mantık biriminin (ALU'nun) fonksiyonel birimlerle, W yazmacı başta olmak üzere ilişkisini ve diğer birimlerle bilgi alış-verişini;

16F87X serisi mikrodenetleyicilerde, komutun sonuna konan 1 veya 0 sayısıyla (d), sonucun W' de ya da başka bir yazmaçta tutulacağı mikroişlemciye bildirilir. Bunun hangi komutlar için ve nasıl yapıldığı komut setinde anlatılmıştır.



Şekil 6.8. ALU'nun çalışması

2. Aritmetik ve mantık biriminin (ALU) çalışması sonucunda, STATUS durum yazmacı ile ilişkisini çalıştırılan komuta bağlı olarak ALU, Carry (C), Digit Carry (DC) ve Zero (Z) bitlerini değiştirir.
3. Program sayacının; yığın, program, veri belleği ile çalışmasını,
4. Erişim türlerinde kullanılan yazmaçları,
5. Osilatör yani zamanlayıcılarla, komut yürütme ilişkileri,
6. Adres yolunun (Program bus) komutlar için nasıl çalıştığı ve veri yollarının (data bus) veri belleği ve giriş-çıkış birimleri ile nasıl işlem gördüğü,
7. Çevresel giriş-çıkış birimleri, portlar ve pin tanımları gibi sayılabilecek pek çok önemli konu izlenebilmektedir.

C) 16F87x Mikrodenetleyicisi İç Mimarisinin Temel Özellikleri

- Veri yolu (databus) 8 bittir.
- 32 Adet SFR (Special Function Register) olarak adlandırılan özel işlem yazmacı vardır ve bunlar statik RAM üzerindedir.
- 8 Kx14 sözcüğe (words) kadar artan FLASH program hafızasına sahiptir (1 milyon kez programlanabilir).

- 368x8 Byte'a kadar artan VERİ hafızasına (RAM) sahiptir.
- 256x8 Byte'a kadar artan EEPROM VERİ hafızası vardır.
- 14 Kaynaktan kesme yapabilir.
- Donanımsal yığın derinliği 8 'dir.
- Doğrudan, dolaylı ve bağıl (relatif) adresleme yapabilir.
- Enerji verildiğinde sistemi resetleme özelliğine (Power-on Reset, POR) sahiptir.
- Enerji verilmesi ile zamanlayıcı (Power-up Timer, PWRT) ve Osilatörü (Oscilator Start-up Timer, OST) başlatma
- Güvenilir işlemler için yonga içindeki RC osilatörü ile zamanlama (Watch-dog Timer-özel tip zamanlayıcı, WDT)
- Program kodunun güvenliğini sağlayabilme
- Enerji tasarrufu sağlayan uyku (SLEEP) modu
- Seçimli osilatör özellikleri
- Düşük güçle, yüksek hızla erişilebilen, CMOS/FLASH/EEPROM teknolojisi
- Tamamen statik tasarım
- 2 pin vasıtası ile devre için seri programlanabilme (ICSP) özelliği
- Yalnız 5 V girişle, devre içi seri programlanabilme yeteneği
- İki pin vasıtası ile devre içi hata ayıklama (In-Circuit Debugger) özelliği
- İşlemcisinin program belleğine, okuma/yazma özelliği ile erişimi
- 2 - 5 Volt arasında değişen geniş işletim aralığı
- 25 mA 'lik (Sink/Source) giriş/çıkış akımı
- Geniş sıcaklık aralığında çalışabilme özelliği
- Düşük güçle çalışabilme değerleri :

  < 0.6 mA, 3V, 4 Mhz.

  < 20 μA, 3V, 32 kHz

  < 1 μA, standby akımı

D) Çevresel Özellikler

- Timer0 (TMR0) : 8 bit ön-ölçekleme ile 8 bitlik zamanlayıcı/sayıcı
- Timer1 (TMR1) : ön-ölçeklemeli 16 bitlik zamanlayıcı/sayıcı; harici bir clock ile uyuma modunda iken arttırılabilir.
- Timer2 (TMR2) : 8 bitlik peryod kayıtçı, ön-bölme ve son bölme ile 8 bitlik zamanlayıcı
- İki Capture, Compare ve PWM modülü (CCP1,2)
- Capture 16 bitlik, maksimum hızı 12.5 ns
- Compare 16 bitlik, maksimum hızı 200ns
- PWM maksimum hızı 10-bit

- 10 bit çok kanallı (8) A/D dönüştürücü
- SPI (Master mod) ve I^2C (Master Slave) ile birlikte Senkron seri port (SSP)
- 9 bit adres belirlemeli USART/SCI (Üniversal senkon-asenkron alıcı/verici)
- 8 bit genişlikte ve dış RD, WR, CS kontrolleri ile Paralel Slave Port (PSP)
- BOR Reset (Brown-out Reset) özelliği

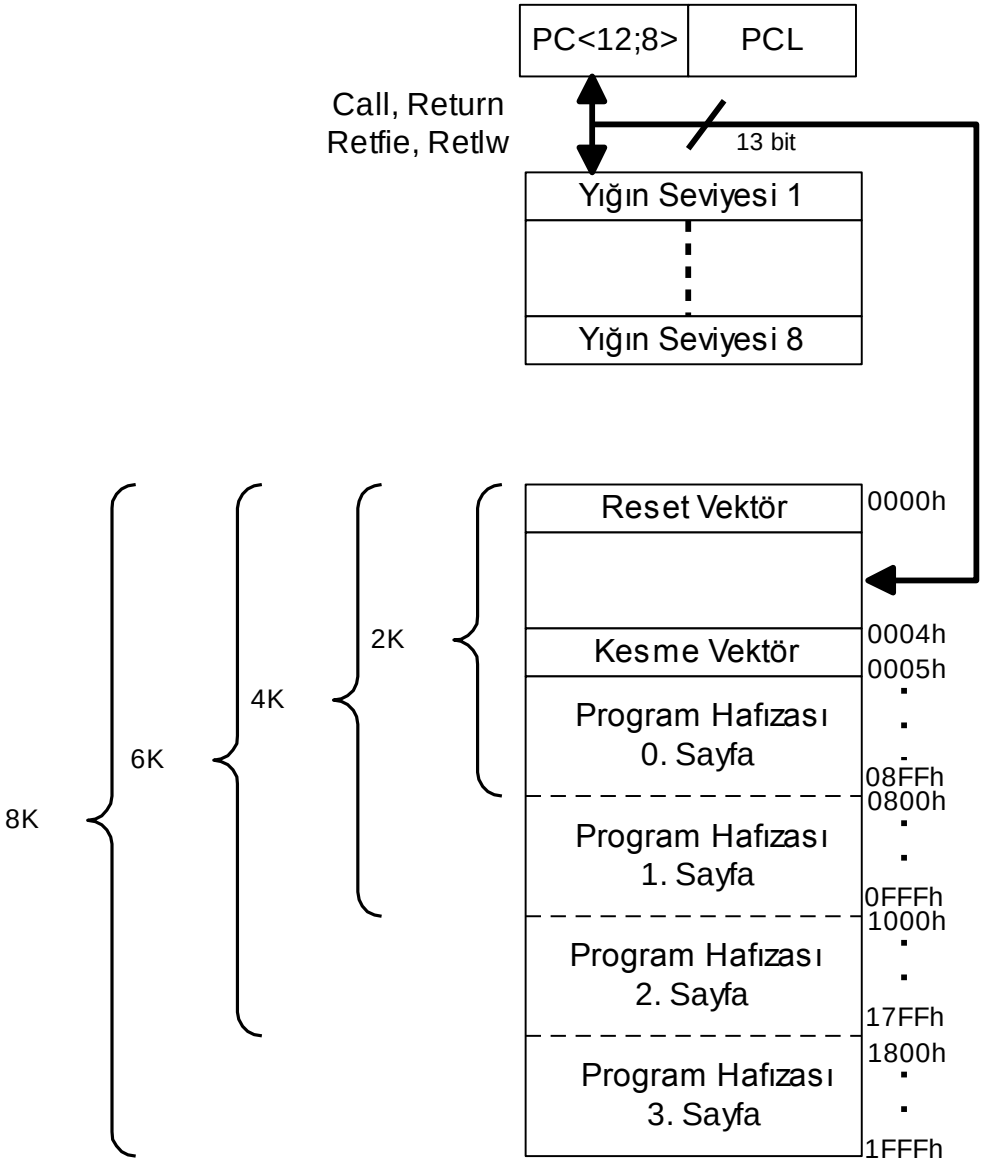
E) PIC Hafıza Organizasyonu

PIC16F877 mikrodenetleyicisinin hafıza yapısı iki ayrı bellek bloğundan oluşur:

- Program belleği
- RAM veri belleği (Kayıtçı Dosya Belleği-Regiter File Memory)

Program Belleği

Çalıştırılacak komut kodlarını tutmak için kullanılır. Her bir satırı 14 bit uzunluğundadır. Her komut 14 bitlik bir sözcük uzunluğuna sahiptir. PC 13 bitlik adres uzunluğuna sahip olduğu için 8Kxsözcük uzunluğuna kadar adresleyebilir.



Şekil 6.9: Adresleme ve Program belleği

Burada:

1. 000_H - 004_H adresleri özel amaçlar (reset, kesme,…, gibi) için ayrılmıştır.
2. Hafıza bloğu dört sayfadan (0000_H – $07FF_H$, 0800_H – $0FFF_H$, 1000_H – $17FF_H$, 1800_H – $1FFF_H$) oluşmaktadır.

Veri Belleği (RAM File Register)

Veri belleği kendi içinde, bank adı verilen sayfalara bölünmüştür. Bunların her birinin başında, özel fonksiyonlu kayıtçı (Special Function Register-SFR) alanı ve daha sonra da genel amaçlı kayıtçı (General Purpose Registers-GPR) alanı bulunur. Özel işlem yazmaçları, mikrodenetleyicinin işletimini kontrol eder ve bir işlemin sonucunu öğrenebileceğimiz, özel durum bitlerini bulundurur :

- Örneğin STATUS yazmacının, 5 ve 6. bitleri olan RP0, RP1 adlı bitler; bank seçimi bitleri olarak, bu bellek bölümlerini seçmede kullanılır. Her bank $07F_H$ adresine dek genişletilmiştir (128 Byte).

RP1:RP0	Bank
00	0
01	1
10	2
11	3

Şekil 6.10.: Bank seçimi

- Bankların başındaki adresler, özel işlem yazmaçlarına ayrılmıştır. Özel işlem yazmaçlarının altındaki adresler ise genel amaçlı yazmaçlara ayrılmıştır. Bunlar statik RAM üzerindedir.
- Bazı özel işlem yazmaçları bir banktan, daha çok bankta yer alır. Bu yöntem, erişimi hızlandırma amaçlı olup, çok kullanılan yazmaçların görüntüsü ayna gibi diğer banklara yansıtılmıştır. Böylece bu yazmaçlara erişmek için sık sık bank değiştirilmesi gereği ortadan kaldırılmış ve programlamaya kolaylık sağlanmıştır.

Veri belleği olarak kullanılmak üzere, kılıfın içerisinde bir de EEPROM bellek alanı vardır. Bu bellek, diğer veri belleği gibi doğrudan adreslenemez. EEPROM veri belleğine dolaylı erişilir. Toplam 92 byte olan bu belleğe nasıl erişebileceğimizden ilerde daha ayrıntılı bahsedeceğiz.

Aşağıdaki şekilde mikrodenetleyicilerde bulunan önemli bir kısım özel işlem kayıtçıları gösterilmiştir :

Kayıtçı Adresi	**BANK0 Kayıtçı adı**	**Kayıtçı Adresi**	**BANK1 Kayıtçı adı**
00_H	INDF	80_H	INDF
01_H	TMR0	81_H	OPTION
02_H	PCL	82_H	PCL
03_H	STATUS	83_H	STATUS
04_H	FSR	84_H	FSR
05_H	PORTA	85_H	TRISA
06_H	PORTB	86_H	TRISB
07_H	PORTC	87_H	TRISC
08_H	EEDATA	88_H	EECON1
09_H	EEADR	89_H	EECON2
$0A_H$	PCLATH	$8A_H$	PCLATH
$0B_H$	INTCON	$8B_H$	INTCON
$0C_H$	(GPR)	$8C_H$	(GPR)
$7F_H$		FF_H	

Şekil 6.11. Bellek Haritası

F) Kayıtçıların İşlevleri

INDF (Indirect File Register): Dolaylı adresleme yazmacıdır. Birbiri ardı sıra yapılacak erişim işlemlerinde, GPR-Genel amaçlı yazmaçlarla (statik RAM alanının) kullanımı hızlandırılır ve yazılacak programı küçültür.

TMR0 (Timer): Mikrodenetleyici içinde bulunan zamanlayıcı ve sayaç olarak çalıştırılan bölümü denetleyen yazmaçtır.

PCL (Program Counter Low Byte): Bir sonra çalıştırılacak komutun program belleğindeki adresini tutar.

STATUS: Mikrodenetleyici içindeki aritmetik işlem birimi (ALU), işlem sonuçlarına ait bazı bilgileri durum yazmacında tutar. Bank seçme bitleri de bu yazmaçtadır. Programa isteğine göre bu birimleri yönlendirir.

FSR (File Select Register): Dolaylı adreslemede INDF ile birlikte kullanılır. Mikrodenetleyicinin içindeki RAM adresinde yapılacak işlemlerde, RAM adresini tutar. Bu durumda INDF 'ye yazılacak her veri, aslında adresi FSR' de bulunan RAM' a yazılmıştır. Aynı şekilde INDF den okunan veri de adresi FSR de bulunan RAM dan okunmuştur.

PORTA - PORTE: Portlar, mikrodenetleyicinin dış dünyadan bilgi alması ve kendi dışındaki devrelere veri aktarabilmesi amacıyla kullanılır. P1C16F877 nin beş portu vardır. A Portu 6 bit genişliğindedir. B, C, D portları 8 bit, E portu ise 3 bit genişliğindedir.

TRISA - TRISE: Portların yönünü (yongaya veri girişi mi, yoksa yongadan veri çıkışı mı yapılacak?) belirleyen yazmaçlardır. Eğer portların herhangi bir pininden mikrodenetleyici dışına veri gönderilecekse, önce ilgili portun yön yazmacının aynı numaralı biti, "0" yapılır. Eğer o pinden mikrodenetleyiciye veri girilecekse, yine önceden, o portun yön yazmacının aynı numaralı biti "1" yapılır. Özetle ilgili TRIS yazmacı pini çıkış için "0", giriş için "l" yapılır.

EEDATA ve EEADR: Mikrodenetleyici içindeki EEPROM (kısaca E^2 veri belleğine ulaşmakta kullanılırlar. Sonuçta EEDATA yazmacındaki veri EEADR yazmacında adres numarası bulunan EEPROM belleğine yazılır. Ya da EEADR yazmacında adres numarası bulunan veri, EEPROM veri belleğinden okunarak EEDATA yazmacına getirilir.

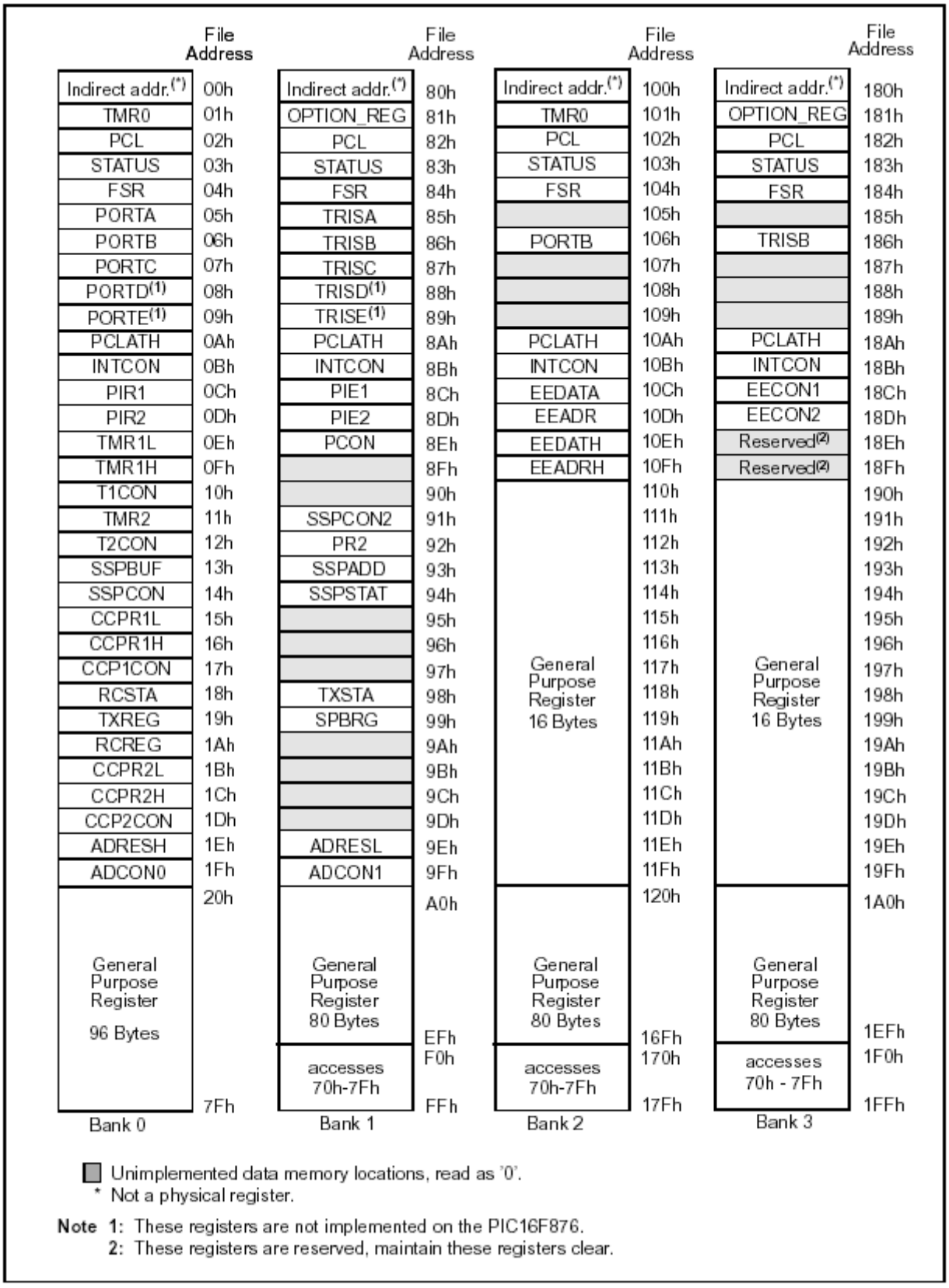
PCLATH: Program sayacının yüksek öncelikli byte yani, üst 5 biti için kullanılır.

INTCON: Kesme (interrupt) işlemlerinde kullanılır.

GPR (General Purpose Register): Genel amaçlı yazmaçların adresleri ilgili şemalarda gösterilmiştir. Programcı buradaki adresleri istediği gibi, kendi değişkenleri için kullanabilir. Bu adresleri isterse programının içinde, aşağıdaki örnekte görüldüğü gibi adlandırabilir.

ISI_1 EQU '20_H' : GPR alanındaki '20_H' adresine ISI_1 adı verildi.

ISI_2 EQU '21_H' : GPR alanındaki '21_H' adresine ISI_2 adı verildi.



Şekil 6.12. PIC16F877 Kayıtçı dosya haritasının tam listesi

SFR adıyla tanımlanan özel işlem yazmaçların tümü, adresleri, adları, sırasıyla bütün bitlerine verilen isimleriyle ve power-on-reset ile brown-on-reset durumunda alacağı değerleriyle birlikte aşağıdaki tablolarda verilmiştir. Bu yazmaçların bir kısmı ilerdeki bölümlerde anlatılacaktır.

Reset Ve Kesme Durumu

Reset (başlama) vektörü: Enerji uygulandığında (Power-on) mikroişlemcinin içinde veya dışında olan bir elektronik devre ile yeniden başlatılması olayı reset işlemidir. Bu devre "power-on reset" adı ile kılıf içerisine yerleştirilmiştir. Çalışmaya başlatılan mikroişlemci, kendi program sayacını özel bir sayı ile yükler. İşte bu sayı, o mikroişlemci için, reset vektör adresidir. Örnek olarak, 16F877 reset vektör adresi 0000_H 'dır.

Kesme (Interrupt) vektörü: Mikroişlemci program belleğindeki programı çalıştırırken, sırası belirsiz, acilen yapılması gerekli yordamları da çalıştırabilir. Sırası ve ne zaman ortaya çıkacağı bilinemeyen bu işleri yapmak için mikroişlemci, bir yolla dışarıdan veya kendi içinden uyarılmalıdır. Gelen uyarıdan mikroişlemcinin bazı birimleri etkilenir. Bu birimlerden biri olan program sayacına, özel bir sayı yüklenir. Bu sayı, o mikroişlemcinin kesme (interrupt) vektör adresidir. Örnek olarak, 16F877 için kesme vektörünün adresi 0004_H 'dır. Kesme sırasındaki uyarıdan etkilenen diğer birim, yığındır. Yığın, program içinde bir altpogram kullanıldığında, bu alt programdan, asıl program blokuna dönülecek adresi tutar. Kesme de bir altprogram gibi ele alınır. Kesmeye sapıldığında (kesmenin bir çağırma komutu yoktur, herhangi bir anda devreye girebilir), kesme bölümünden sonra dönülecek adres PC' den yığına yerleştirilir. Daha sonra kesme yordamının komutları işlenir. Kesmeden çıkış komutu olan RETFIE, altprogramdan çıkış komutu RETURN gibi çalışır. RETFIE komutu ile, programda dönülecek yerin adresi yığından alınıp, PC' ye geri yüklenir. Böylece kesmeden sonra, program bloğu içinde işlemeyi bıraktığı yere döner ve kalan komutları işlemeye devam eder.

İç içe kullanılan altprogramların en 'çok sekiz olabileceğini söylemiştik. Bunlara kesme bölümleri de dahildir. Kesmeleri dahil etmezseniz, yığın taşmasına neden olursunuz. Yığın taşması oluştuğunda bizi uyaracak, herhangi bir uyarı-flag yazmacı bulunmamaktadır.

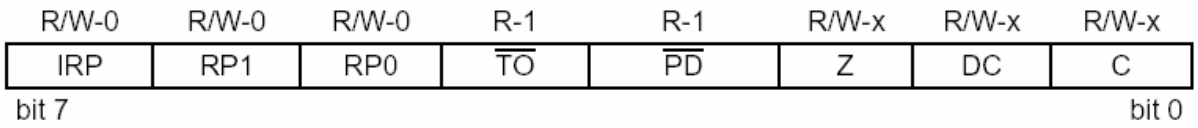
G) Durum (Status) Kayıtçısı

STATUS kayıtçısı ALU biriminin, aritmetik işlem sonucundaki durumunu, CPU test durumlarını ve veri belleğine ait küme (bank) seçme bitlerini tutar. Herhangi bir kayıtçı gibi, STATUS da bir komuta hedef olabilir. Yani, içeriği okunabilir, değiştirilebilir. Ancak, $\overline{TO}$ ve $\overline{PD}$ isimli bitler sadece okunabilir, değiştirilemez.

Eğer, bu kayıtçının içeriği CLRF STATIJS komutuyla, silinmek istenirse; sadece üst üç bit 0 olur. Bu komut sonunda STATUS 'un içeriği 000d d1dd değerini alır. Burada d:değişmeyen anlamındadır.

BCF, BSF, SWAPF, MOVWF komutları ile $\overline{TO}$ ve $\overline{PD}$ bitleri hariç, diğer bitlerin içeriği değiştirilebilir.

STATUS kayıtçısının Kayıtçı dosyasındaki adresleri: 03_H, 83_H, 103_H, 183_H



Şekil 6.13: STATUS Kayıtçısı

Bit 7 IRP: Kayıtçı Bank Seçme Biti (dolaylı adreslemede kullanılır)

0 = Bank 0, 1 (00_H - FF_H)

1 = Bank 2, 3 (100_H - $1FF_H$)

Bit 6-5 RP1: RP0: Kayıtçı Bank Seçme Biti (doğrudan adreslemede kullanılır).

Her bir bank 128 byte dir.

00 = Bank 0 (00_H - $7F_H$)

01 = Bank 1 (80h - FF_H)

10 = Bank 2 (100_H - $17F_H$)

11 = Bank 3 (180_H - $1FF_H$)

Bit 4 $\overline{TO}$**:** Süre aşımı (Time-out) biti

0 = WDT süre aşımı gerçekleşmiş ise "0" olur.

1 = Power-up, CLRWDT veya SLEEP komutu işlemlerinden sonra, güç verme durumuna geçilmiş ise "1" olur.

Bit 3 $\overline{PD}$**:** Güç kesme (Power-down) biti

0 = SLEEP komutu çalıştırılınca

1 = CLRWDT komutu çalıştırılınca veya güç verme durumunda

Bit 2 Z: Sıfır biti

0 = Aritmetik veya lojik işlemin sonuçu sıfır değil ise

1 = Aritmetik veya lojik işlemin sonuçu sıfır ise

Bit 1 DC: Önemsiz Basamağın tasma / $\overline{borç}$ (carry / $\overline{borrow}$) biti

(ADDWF, ADDLW,SUBLW,SUBWF komutları için)

0 = İşlem sonucunda düşük 4-bitlik kısımdan taşma (carry) eldesi yoksa

1 = İşlem sonucunda düşük 4-bitlik kısımdan taşma (carry) eldesi varsa

Bit 0 C: Önemli Basamağın tasma / $\overline{borç}$ biti

(ADDWF, ADDLW,SUBLW, SUBWF komutları için)

0 = İşlem sonucunda en önemli bitte taşma yoksa

1 = İşlem sonucunda en önemli bit taşarsa

NOT: Bit 0 ve **1** de; ödünç alma (borrow) işlemleri için ters kutup kullanılmıştır. Çıkarma (SUB) ve döndürme (RLF, RRF) işlemlerinde bunun etkisi anlatılacaktır.

H) Seçenek Kayıtçısı (OPTION_REG)

Option kayıtçısı, okunabilir ve yazılabilir bir kayıtçıdır. Kapsamında TMR0/WDT zamanlayıcılarının konfigürasyon bitleri, dış kesme denetim bitleri, TMR0 zamanlayıcısı kesme denetim bitleri ve PORTB için çekme (pull-enable) dirençlerinin kullanılmasını sağlayan bit bulunur. OPTION kayıtçısının Kayıtçı dosyasındaki adresleri: 81_H, 181_H

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

Şekil 6.14: OPTION Kayıtçısı

Bit 7 RBPU: PORTB, çekme (pull-up) işlemini mümkün kılma biti

0 = PORTB çekme aktif ise

1 = PORTB çekme pasif ise

Bit 6 INTEDG: Kesme kaynağının tetikleme kenarının seçim biti

0 = RB0/INT uçunun düşen kenarında kesme

1 = RB0/INT uçunun yükselen kenarında kesme

Bit 5 TOCS: TMR0 saat kaynağını seçme biti

0 = Dahili komut çevrim clocku kullanılır (CLKOUT)

1 = RA4/TOCK1 pininden (uçundan) gelen darbeler clock kaynağı olur

Bit 4 TOSE: TMR0 kaynak kenarı seçme biti (Eğer TOCS = 1 ise)

0 = RA4/TOCK1 pininden gelen her yükselen kenar için bir artırılır

1 = RA4/TOCK1 pininden gelen her düşen kenar için bir artırılır

Bit 3 PSA: Önbölücü / önölçekleme yapılacak birimi seçme biti

0 = Önbölücü TMR0 modülü için ayrılır

1 = Önbölücü WDT için ayrılır

Bit 2,1,0; PS2, PS1, PS0: Önbölücü oranı seçme bitleri

PS2 PS1 PS0 Bit Değerleri	TMR0 Bölme Oranı	WDT Bölme 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
011	1:128	1:64
111	1:256	1:128

Şekil 6.15. Ön bölme seçme bitleri

I) Kesme Kayıtçısı (INTCON)

INTCON kayıtçısı, okunabilir ve yazılabilir bir kayıtçıdır. Kapsamında TMR0/WDT kayıtçılarının taşma uyarı bitleri, RB port değişim ve dış kesme (RB0/INT pin interrupt) denetim bitleri, TMR0 kesme denetim bitleri bulunur. Kesme bitleri şöyle kullanılır;

1-) Bir kesme durumu oluşturulacaksa, programcı önce GIE (Global Interrupt Enable) (INTCON <7>) bitini set eder.

2-) Bu bit set edildikten sonra, programcının kullanmak istediği kesme veya kesmeler aktifleştirilir. Programcı kullanmak istediği her kesme için, ilgili kesmeyi aktifleştirme bitinide kullanmalıdır (Aktifleştirme bitlerinin adlarının sonunda (Bit 6-5, 4-3) "**I**nterupt **E**nable" sözcüklerinin baş harfleri bulunur).

3-) Kesme oluşturulduktan sonra ise kesmeyle ilgili uyarı yada bayrak (Interrupt Flag) bitini programcı kontrol etmelidir.

4-) Programcı kesme ile uyarı/bayrak bitlerini kontrol ederse (kesme yordamında), bunların taşma durumunda "l", aksi halde "0" olduğunu görecektir.

5-) Programcının, program çalıştığı sürece, kesmeyi sürdürebilmesi için, kullandığı kesmeyle ilgili uyarı bitini (Interrupt Flag Bit) kendisinin sıfırlaması gerekir. Kesme uyarısı yada bayrak biti denilen bu bitler programcı tarafından, temizlenmez (sıfırlanmaz) ise bir daha kesme oluşturulamaz. INTCON kayıtçısının Kayıtçı Dosyasındaki adresleri: $0B_H$, $8B_H$, $10B_H$, $18B_H$

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	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
bit 7							bit 0

Şekil 6.16. INTCON Kayıtçısı

Bit 7 GIE: Bütün kesmeler geçerli (Global Interrupt Enable) biti

0 = Kesmelere izin vermez.

1 = maskelenmemiş kesmelere izin verir.

Bit 6 PEIE: Çevresel kesme geçerli biti

0 = Çevresel kesmelere izin vermez.

1 = Maskelenmemiş çevresel kesmelere izin verir.

Bit 5 TOIE: TMR0 taşma kesmesi geçerli biti

0 = TMR0 kesmesine izin vermez.

1 = TMR0 kesmesine izin verir.

Bit 4 INTE: RB0/INT (pininden gelen) dış kesme geçerli biti

0 = RB0/INT dış kesmeye izin vermez.

1 = RB0/INT dış kesmeye izin verir.

Bit 3 RBIE: RB Port değişim kesmesi geçerli biti

0 = RB port değişim kesmesine izin vermez.

1 = RB port değişim kesmesine izin verir.

Bit 2 TOIF: TMR0 taşma kesmesi bayrak biti

0 = TMR0 kayıtçısı taşmadı.

1 = TMR0 kayıtçısı taştı (taştıktan sonra program içinden temizlenir).

Bit 1 INTF: RB0/INT dış kesme bayrak biti

0 = RB0/INT dış kesme yok.

1 = RB0/INT dış kesme oldu (program içinden temizlenir).

Bit 0 RBIF: RB Port değişim kesmesi bayrak biti

0 = RB4:RB7 pinlerinin hiçbiri durum değiştirmedi.

1 = RB4:RB7 pinlerinin en az biri durum değiştirdi

(Programda kontrol edilir).

J) Çevresel Kesme Kayıtçısı (PIE1)

PIE1, çevresel kesmelerle ilgili bitleri içeren bir kayıtçıdır. Bir çevresel kesmenin geçerli olabilmesi için, PEIE (INTCON <6>) biti de set edilmelidir. PIE1 kayıtçısının Kayıtçı Dosyasındaki adresi: $8C_H$

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
bit 7							bit 0

Şekil 6.17. PIE1 Kayıtçısı

Bit 7 PSPIE: Paralel Slave Port (PSP) okuma/yazma kesmesi geçerlilik biti

0 = PSP R/W kesmesine izin verilmez.

1 = PSP R/W kesmesine izin verilir.

Bit 6 ADIE: A/D çevirici kesmesi geçerlilik biti

0 = A/D çevirici kesmesine izin verilmez.

1 = A/D çevirici kesmesine izin verilir.

Bit 5 RCIE: USART alma (receive) kesmesi geçerlilik biti

0 = USART alma kesmesine izin verilmez.

1 = USART alma kesmesine izin verilir.

Bit 4 TXIE: USART gönderme (transmit) kesmesi geçerlilik biti

0 = USART gönderme kesmesine izin verilmez.

1 = USART gönderme kesmesine izin verilir.

Bit 3 SSPIE: Senkron Seri Port (SSP) kesmesi geçerlilik biti

0 = SSP kesmesine izin verilmez.

1 = SSP kesmesine izin verilir.

Bit 2 CCP1IE: CCP1 kesmesi geçerlilik biti

0 = CCP1 kesmesine izin verilmez.

1 = CCP1 kesmesine izin verilir.

Bit 1 TMR2IE: TMR2 ile PR2 uyum kesmesi geçerlilik biti

0 = TMR2 ile PR2 uyum kesmesine izin verilmez.

1 = TMR2 ile PR2 uyum kesmesine izin verilir.

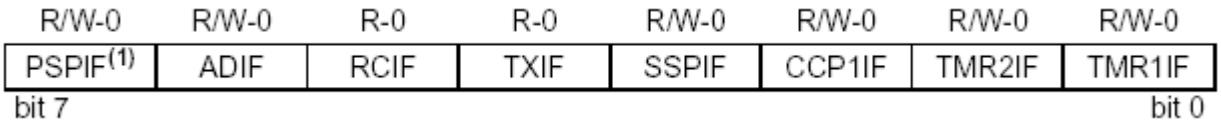
Bit 0 TMR1IE: TMR1 taşma kesmesi geçerlilik biti

0 = TMR1 taşma kesmesine izin verilmez.

1 = TMR1 taşma kesmesine izin verilir.

K) Çevresel Kesme Kayıtçısı (PIR1)

PIR1 kayıtçısı çevresel kesmelerle ilgili uyar bitlerini taşıyan bir kayıtçıdır. Programcı kesmenin oluşup oluşmadığını ve kesmeyle ilgili oluşan olayları, bu uyarı bitlerini denetleyerek anlar. Kesmenin tekrar oluşturulabilmesi için, ilgili uyarı yada bayrak biti yazılımla temizlenmelidir. PIR1 kayıtçısının Kayıtçı Dosyasındaki adresi: $0C_H$



Şekil 6.17. PIR1 Kayıtçısı

Bit 7 PSPIF : Paralel Slave Port okuma/yazma kesme uyarısı geçerlilik biti

0 = Okuma yada yazma yok.

1 = Okuma yada yazma işlemi gerçekleşti (yazılımda temizlenmeli).

Bit 6 ADIF : A/D Çevirici kesmesi uyarı biti

0 = A/D dönüşüm tamamlanmadı.

1 = A/D dönüşüm tamamlandı.

Bit 5 RCIF : USART alma kesmesi uyarı biti

0 = USART alma tamponu boş.

1 = USART alma tamponu dolu.

Bit 4 TXIF : USART gönderme kesmesi uyarı biti

0 = USART gönderme tamponu boş.

1 = USART gönderme tamponu dolu.

Bit 3 SSPIF : Senkron seri port (SSP) kesme uyarı biti

0 = SSP kesme şartları sağlanmadı.

1 = SSP kesme şartları sağlandı (kesme hizmet programından geri

dönmeden önce yazılımla temizlenmeli)

Bit 2 CCP1IF : CCP1 kesmesi uyarı biti : Capture ve Compare modunda kullanılır. PWM modunda kullanılmaz.

0 = TMR1 kayıtçısı capture/compare vuku buldu.

1 = TMR1 kayıtçısı capture/compare vuku bulmadı.

Bit 1 TMR2IF : TMR2 - PR2 uyum kesmesi uyarı biti

0 = TMR2 - PR2 uyum yok.

1 = TMR2 - PR2 uyum var. (yazılımla temizlenmeli)

Bit 0 TMR1IF : TMR1 taşma kesmesi uyarı biti

0 = TMR1 kayıtçısı taşma olmadı.

1 = TMR1 kayıtçısında taşma oldu.(yazılımla temizlenmeli)

L) PIE2 Çevresel Kesme Kayıtçısı

PIE2 kayıtçısı, CCP2 (Capture/Compare/PWM 2) çevresel biriminin kesme bitlerini, SSP (Senkron Seri Port) veri yolu çarpışma (bus-collision) bitini ve EEPROM yazma kesmesi bitini taşır. PIE2 kayıtçısının Kayıtçı Dosyasındaki adresi: $8D_H$

U-0	R/W-0	U-0	R/W-0	R/W-0	U-0	U-0	R/W-0
—	Reserved	—	EEIE	BCLIE	—	—	CCP2IE
bit 7							bit 0

Şekil 6.18. PIE2 Kayıtçısı

Bit 7: Bu bit kullanılmaz, 0 okunur.

Bit 6 Reserved: Bit sonra kullanılmak için ayrılmıştır. Temizlenmelidir (set 0).

Bit 5: Bu bit kullanılmaz, 0 okunur.

Bit 4 EEIE: EEPROM yazma işlem kesmesi geçerlilik biti

0 = EEPROM yazma kesmesine izin verilmez.

1 = EEPROM yazma kesmesine izin verilir.

Bit 3 BCLIE: Çarpışma (Bus collision) kesmesi geçerlilik biti

0 = BUS Çarpışma kesmesine izin verilmez.

1 = BUS Çarpışma kesmesine izin verilir.

Bit 1-2: Bu bitler kullanılmaz, 0 okunur.

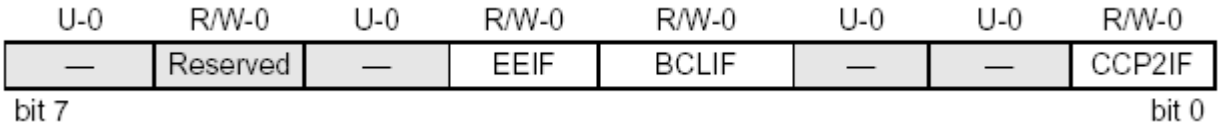
Bit 0 CCP2IE: CCP2 kesme geçerlilik biti

0 = CCP2 kesmesine izin verilmez.

1 = CCP2 kesmesine izin verilir.

M) PIR2 Çevresel Kesme Kayıtçısı

PIE2 kayıtçısı, CCP2 çevresel biriminin kesme bitlerini, SSP çarpışma bitini ve EEPROM yazma kesmesi uyarı bitini taşır. PIR2 kayıtçısının Kayıtçı Dosyasındaki adresi: $0D_H$



Şekil 6.19. PIR2 Kayıtçısı

Bit 7: Bu bit kullanılmaz, 0 okunur.

Bit 6 Reserved: Bit sonra kullanılmak için ayrılmıştır. Temizlenmelidir (set 0).

Bit 5: Bu bit kullanılmaz, 0 okunur.

Bit 4 EEIF: EEPROM yazma işlemi kesme uyarı biti

0 = yazma işlemi tamamlanmadı.

1 = yazma işlemi tamamlandı.

Bit 3 BCLIF: Çarpışma (Bus collision) kesmesi uyarı biti

0 = SSP de çarpışma oldu, (12C Master mod olarak yapılandırılmışsa)

1 = Çarpışma olmadı.

Bit 1-2: Bu bitler kullanılmaz, 0 okunur.

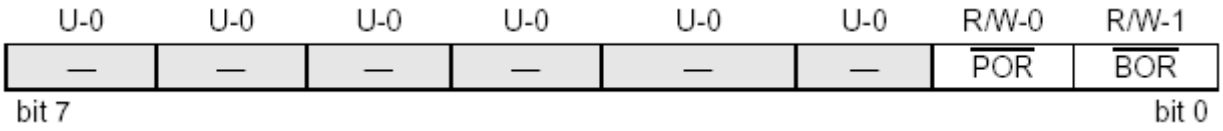
Bit 0 CCP2IF : CCP2 Kesme uyarı biti : Capture ve Compare modunda kullanılır. PWM modunda kullanılmaz.

0 = TMR1 kayıtçısı capture/compare vuku buldu.

1 = TMR1 kayıtçısı capture/compare vuku bulmadı.

N) PCON Güç Kaynağı Kontrol Kayıtçısı

Güç kontrol kayıtçısı PCON, yazılımda ve reset durumlarında kullanılır. Reset durumları; devrenin dışardan MCLR ile, gerilim yada akımın aşırı düşme ve yükselmesi Brown-Out Reset (BOR), Wath-Dog Timer, ve son olarak Power-on Reset (POR) durumlarında kullanılabilir. BOR biti, Power-on Resette bilinemez. Reset sonrasında "1" yapılmalıdır ki, bir sonraki BOR durumu öğrenilebilsin. BOR durumunun başka türlü öğrenilebilmesi mümkün değildir. BOR biti temizlendiğinde, Brown-out devresini kullanımdan kaldırır (disable). PCON kayıtçısının Kayıtçı Dosyasındaki adresi: $8E_H$



Şekil 6.20: PCON Kayıtçısı

Bit 2-7: Bu bitler kullanılmaz, 0 okunur.

Bit 1 POR: Power-On Reset durum biti

0 = POR oluştu (POR oluştuktan sonra yazılımla set edilmeli).

1 = POR oluşmadı.

Bit 0 BOR: BOR durum biti

0 = BOR durumu var (BOR oluştuktan sonra yazılımda set edilmeli).

1 = BOR durumu yok.

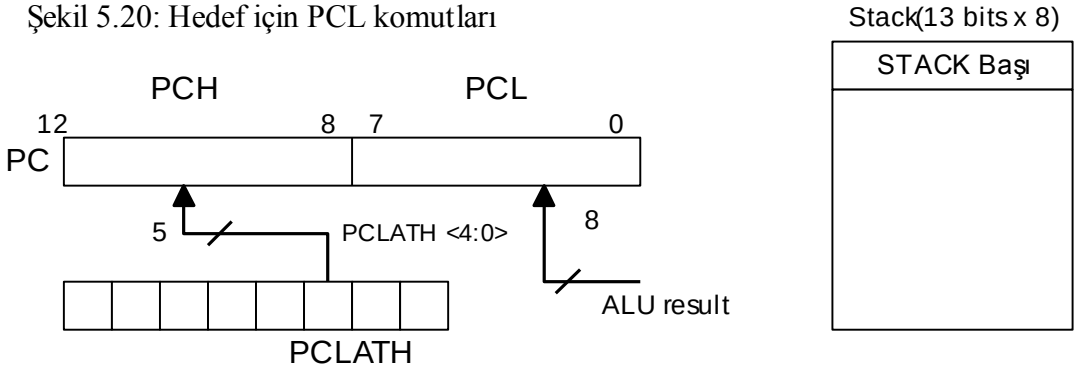
O) PCL ve PCLATH Adres Kayıtçıları

Program Counter-PC olarak adlandırılan adresleme kayıtçısının 13 bit genişlikte olduğunu söylemiştik. Bunun düşük öncelikli byte 'ı PCL kayıtçısından gelir. Üstteki bitler ise, PC <12:8> arasındaki 5 bittir, bunlar PCLATH kayıtçısından alınır. PCL okunabilir ve yazılabilir bir kayıtçıdır. Ancak üst bitleri (PCH) doğrudan okunamaz. Dolaylı olarak PCLATH yoluyla yazılabilir veya okunabilir. RESET durumunda üst bitler temizlenir. Aşağıda şekilde PC kayıtçısının değişik durumlarda nasıl yüklendiği gösterilmiştir.

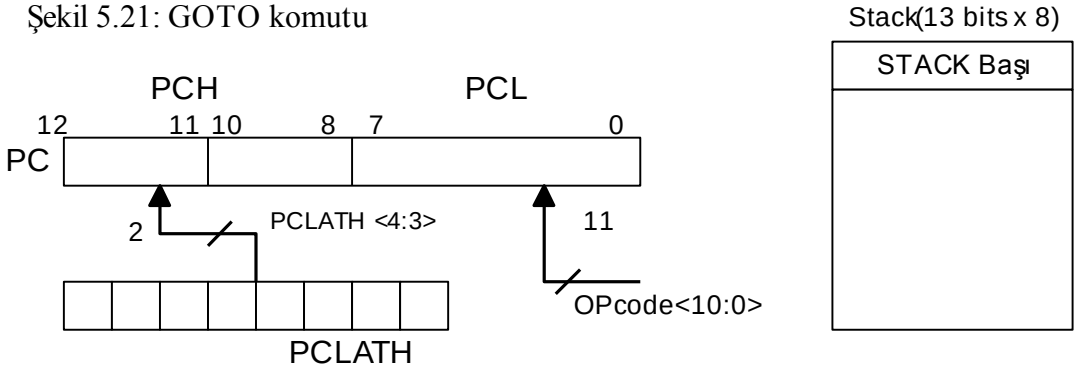
Bunlardan ilki PC yazmacının düşük öncelikli (low) byte ve yüksek öncelikli (high) byte'larının nereden ve nasıl yüklendiğini, ikincisi ise GOTO komutunda nasıl yüklendiğini açıklar. CALL komutunda ise PCL, PCH ve PCLATH ilişkisini gösterdiği gibi, yığınla PC ilişkisini de vurgulanmaktadır. CALL komutu, yığının her zaman en tepesine, PCL yazmacının içindeki adres değerini yazar.

RETURN, RETFIE ve RETLW komutları ise yığının en tepesindeki elemanın içeriğini PCL 'ye aktarır. Sayfa (bank) numaralarının PCLATH kayıtçısından PC 'ye aktarılabildiğini program yazarken de unutmamalıyız.

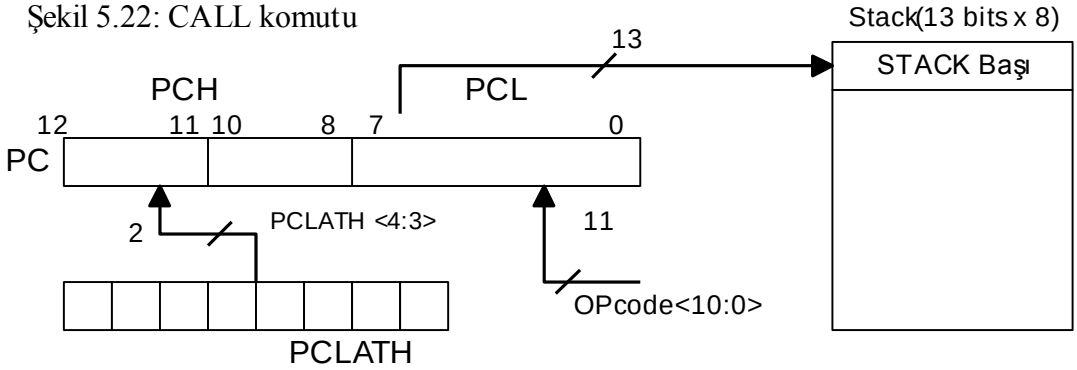
Şekil 5.20: Hedef için PCL komutları



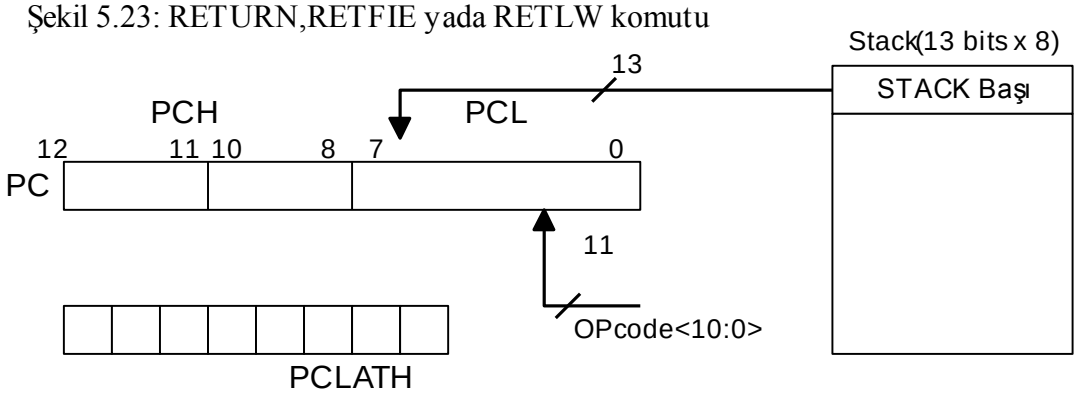
Şekil 5.21: GOTO komutu



Şekil 5.22: CALL komutu



Şekil 5.23: RETURN,RETFIE yada RETLW komutu



Şekil 6.21. Program Sayacının Kullanım Biçimleri

PCLATH kayıtçısının içeriği, altyordama girildikten sonra sabit kalır, bir RETURN yada RETFIE benzeri komut gelse de değişmez. Programcı, CALL veya GOTO komutlarından önce, PCLATH kayıtçısını güncellemelidir. PCH daima PCLATH kayıtçısı yoluyla güncellendiğinden

(tersi yapılamaz), altyordam veya gidilen kesimin hangi sayfada olduğu, aşağıdaki örneğe benzer bir yolla belirtilmelidir.

```
ORG 0x500          0. Sayfada
Bcf PCLATH, 4;     PCLATH kayıtçısının 4. biti temizlendi
Bsf PCLATH, 3;     PCLATH'in 3. biti set edildi, 1. sayfaya geçildi.
                   (800H-FFF H adres aralığı)
call SUB1_P1;      1. sayfadaki altyordam çağrıldı
….
ORG 0x900;  (800 H -FFF H), 1. Sayfada
SUB1_P1
….                 Altyordam 800 H ile FFF H aralığına yerleştirildi.
RETURN;            return'den sonra 0. sayfaya dönülecek.
```

Hesaplanmış GOTO (Computed goto)

Computed goto (ADDWF PCL), PC 'ye PCL 'nin eklenmesiyle oluşur. Eğer computed goto yöntemiyle çalışacaksanız, bellek sınırı içerisinde kalmaya özen göstermelisiniz (her blok 256 byte ile sınırlıdır).

Ö) Yığın (Stack)

Yığın 8 elemanlıdır. Elemanları 13-bitliktir ve donanımın bir parçasıdır. Veri veya program alanlarında yer almaz. Yığın göstergesi (pointer) yazılabilir ve okunabilir değildir. Yığın, dairesel bir dizin gibi çalışır. Eğer iç içe 9. kez altprogram çağırıldıysa; 9. adres, yığının ilk elemanının üzerine yazılacaktır. Bu durumda stack overflow denen, yığın taşması oluşur. PIC' lerde yığın taşmasını denetleyebileceğiniz bir uyarı biti bulunmadığından, bunu kendi yazılımınız yoluyla kontrol etmelisiniz.

Yığın işlemi komutları POP ve PUSH' tur. Her PUSH işleminde (CALL komutuyla ve kesme devreye girdiğinde), yığının en tepesindeki adrese, PC' nin içeriği yüklenir. Her POP işleminde (RETURN, RETFIE, RETLW komutlarında) yığının en tepesindeki adres PC' nin içine geri yüklenir. Programcının yığına erişmesi ve POP, PUSH işlemlerini yapabilmesi olanaksızdır.

Yığın, LIFO (FILO) son giren ilk çıkar tekniğiyle çalışır.

P) INDF ve FSR Dolaylı Erişim Kayıtçıları

INDF, fiziksel bir kayıtçı değildir. Mikrodenetleyicideki RAM adresini tutar. INDF 'e yazılan her veri, adresi FSR 'de bulunan RAM 'a yazılır. INDF 'ten okunan veriler de adresi FSR 'de bulunan RAM 'dan okunmuştur.

Aşağıda örnek olarak, RAM 'da 20_H - $2F_H$ arasındaki yerleri, dolaylı adresleme kullanarak silme işlemi yapan program bölümü gösterilmiştir:

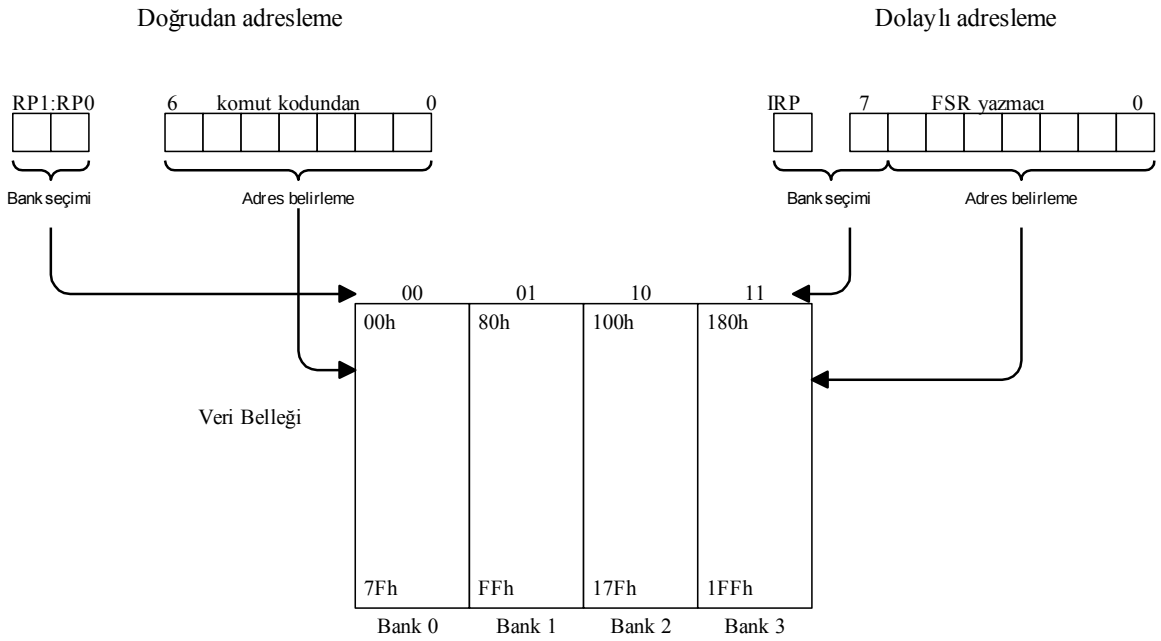
….

Temizle; 20_H-$2F_H$ arasını temizler.

```
        movlw  0x20; Göstergeye başlangıç değerini (adresi) ver .
        movwf FSR;   RAM'a git
Sonraki:
        clrf  INDF;          INDF yazmacını temizle
        incf FSR, F;         Göstergeyi bir arttır.   (d = 1)
        btfss FSR, 4;  Hepsi yapıldı mı?
        goto Sonraki;  Temizlenecek alan bitmedi, sonrakine git.
Başka_Kısıma_Geç;     Temizlenecek alan bitti.
```

….

Şekilde, doğrudan ve dolaylı adresleme nasıl oluşur gösterilmektedir.



Şekil 6.22. Doğrudan ve dolaylı adreslemede INDF ve FSR kullanımı

R) Konfigürasyon Sözcüğü (CPU Özel Fonksiyonları)

PIC 16F87x yongasının, aşağıda sayılan özelliklerinin nasıl kullanılacağı belirlenmelidir. Buna PIC' in konfigürasyonunu belirleme işlemi de denir. Konfigürasyon, MPASM derleyicisinin CONFIG komutuyla belirtilebilir. Konfigürasyon tanımında, her yonganın kendi konfigürasyon sözcüğü (1 word boyunda, 14 bitliktir) içindeki bitlerinin (0-13. bitler) nasıl kullanılacağı, program içinden seçilebilir. PIC16F877 için aşağıda sayılan özellikler, program içinde seçilmediyse, sonradan PIC'e programın yüklenmesi sırasında da seçilebilir. Programlayıcıların da konfigürasyon belirlemek için özel komutları vardır. Konfigürasyon program içinde yapılmadıysa,

programlayıcılar programın yüklenmesinden önce, konfigürasyonun yapılmadığına ilişkin uyarı verir. Konfigürasyon bitlerinde hiçbir değişiklik yapmadığınız takdirde, üretici tarafından belirlenmiş, ön koşullara bağımlı kalınır.

PIC 16F877' nin konfigürasyon bitleri, işlevleriyle aşağıda sayılmıştır.

- Power-on reset (POR)
- Power-up timer (PWRTE)
- Osilatör start-up timer
- BOR (Brown Out Reset)
- Yonga içindeki bir RC osilatör devresi ile belirli bir frekansta çalışması denetlenen WDT(Watchdog timer) birimi
- Kesmeler
- Kod koruma güvenliği
- Id yerleşimleri
- Güç harcamasının azaltılması istendiği durumlar için uyku (sleep) modu
- İsteğe bağlı osilatör seçenekleri: -RC / -XT / -HS / -LS
- Devre içi seri programlama (iki pin ile seri olarak programlanabilme özelliği)
- Devre içi düşük gerilimde programlama,
- Devre içi hata arayıcı (Debugger)

Aşağıda P1C16F877'in, program belleğinde 2007_H adresindeki konfigürasyon sözcükleri, bit açılımı ve değerleri açıklanmaktadır.

CP1	CP0	DEBUG	—	WRT	CPD	LVP	BODEN	CP1	CP0	$\overline{\text{PWRTE}}$	WDTE	F0SC1	F0SC0
bit13													bit0

Şekil 6.23. Konfigürasyon sözcüğünün bit açılımı (adresi: 2007_H)

Bit 12-13 CP0, CP1: Flash Program belleği kod koruma biti

Bit 4-5 : 11 → Kod koruması yok

10 → $1F00_H$ - $1FFF_H$ arası kod korumalı bölge

01 → $1D00_H$ - $1FFF_H$ arası kod korumalı bölge

00 → 0000_H - $1FFF_H$ arası kod korumalı bölge

Bit 11 DEBUG : Devre içi hata arama modu (In-Circuit Debugger Mode)

1= Devre içi hata arama pasif

0= Devre içi hata arama aktif

Bit 10 : Bu bit kullanılmaz, 1 okunur.

Bit 9 WRT: Flash program belleğine yazma biti

1 = Kod korumasız program belleğine EECON denetimi ile yazılabilir.

0 = Kod korumasız program belleğine EECON denetimi ile yazılamaz.

Bit 8 CPD : Veri EE Beleği kod koruma biti

1 = Kod koruması yok

0 = Veri EEPROM belleği Kod korumalı

Bit 7 LVP : Düşük gerilim devre içi seri programlama biti

1 = RB3/PGM (36.pin) Pini PGM işlevlidir, düşük gerilimle programlanabilir.

0 = RB3 sayısal I/O tanımlı, MCLR ye (1.pin) programlama için yüksek gerilim uygulamalıdır.

Bit 6 BODEN : Gerilim alt ve üst limitleri aşarsa, programı yeniden başlatabilen (Brown out Reset Enable) bit

1 = BOR yeniden başlatılabilir

0 = BOR yeniden başlatılamaz

Bit 3 PWRTE: Power-up zamanlayıcı (PWRT) biti

1 = PWRT pasif

0 = PWRT aktif

Bit 2 WDTE : Bekçi köpeği zamanlayıcısı (Watch dog timer, WDT) biti

1 = WDT aktif

0 = WDT pasif

Bit 1-0 FOSC1, FOSC0: Osilatör seçme biti.

11 → RC (direnç kapasite) osilatör seçildi

10 → HS (yüksek hızlı kristal) osilatör seçildi

01 → XT (kristal) osilatör seçildi

00 → LP (düşük güçlü kristal) osilatör seçildi.

6.2. Bölüm Kaynakları

1. *O. Altınbaşak, 2001. "Mikrodenetleyiciler ve PIC Programlama", Atlaş Yayıncılık, İstanbul.*
2. *O. Urhan, M.Kemal Güllü, 2004. "Her Yönüyle PIC16F628", Birsen Yayınevi, İstanbul.*
3. *N. Topaloğlu, S. Görgünoğlu,2003. "Mikroişlemciler ve Mikrodenetleyiciler", Seçkin Yayıncılık, Ankara.*
4. *Y. Bodur, 2001. "Adım Adım PICmicro Programlama",İnfogate.*
5. *www.microchip.com*

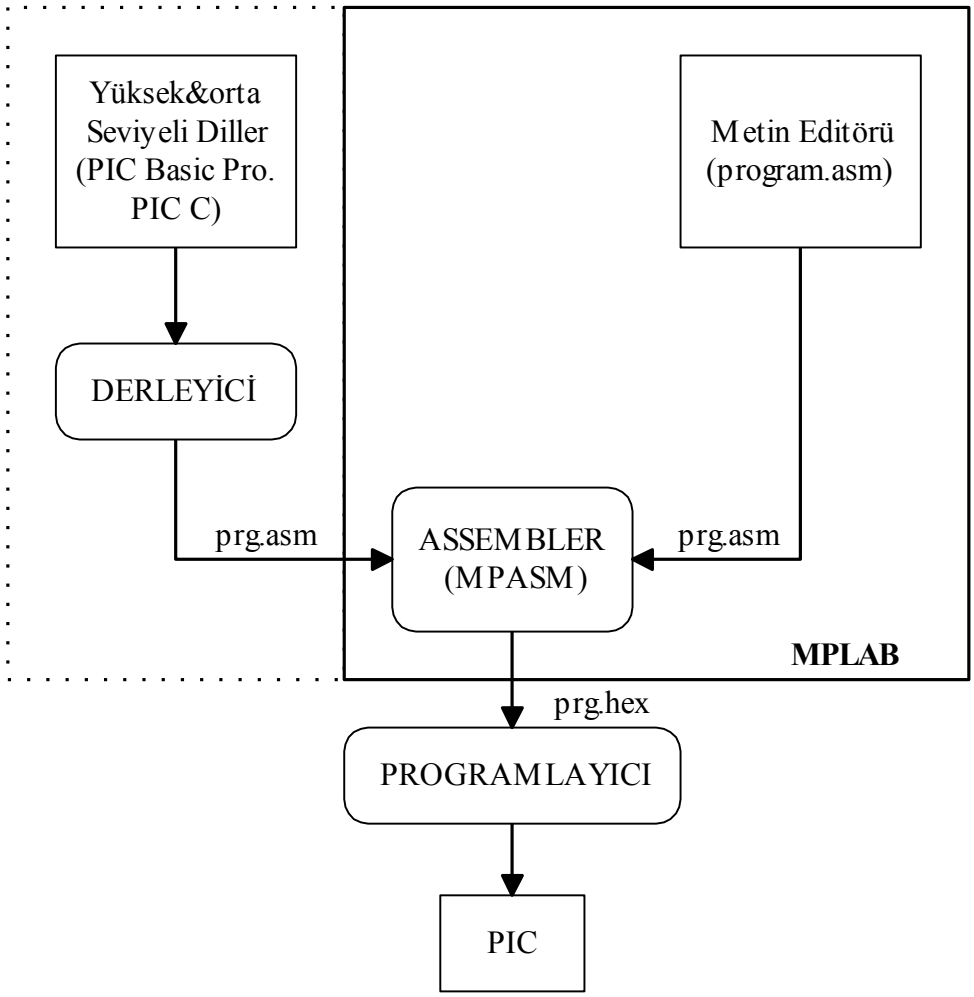
6.3. PIC PROGRAMLAMA VE ASSEMBLY DİLİ

A) PIC Programlama için Gerekenler

PIC serisi mikrodenetleyicileri programlamak için bazı yazılım ve donanım elemanlarına gerek duyulur. Bunlar;

- Kişisel bilgisayarlar (PC)
- Programlama Devreleri
- Metin Editör Programları
- Assembly kodu derleyicileri (assembler)
- Program yükleme yazılımlarıdır.

PIC programlamanın ilk aşamasında program kodlarının yazılması ve PIC'in anlayabileceği makine kodlarına (HEX) yani "0" ve "1" lere dönüştürülmesi gerekmektedir. Bunun için öncelikle bir bilgisayar ve metin editör programına ihtiyaç duyulur. Metin editör programı olarak genellikle Windows işletim sistemi ile birlikte yüklenen "Not defteri" programı kullanılabilir. DOS işletim sistemi ile çalışan bilgisayarlarda ise EDİT programında da aynı işlemler yapılabilir. Yazılan bu assembly program kodları, Microchip tarafından ücretsiz olarak verilen MPSAM programı ile PIC'in işlem yapabildiği HEX kodlarına dönüştürülür. Bu HEX kodlarının PIC'e yüklenmesi için bir programlayıcı devreye ve bu devrenin yazılımına ihtiyaç duyulmaktadır. Programlayıcı devre çeşitleri olarak ta paralel, seri veya USB portlarını kullanan programlayıcılar kullanılmaktadır. Paralel port üzerinden işlem yapacak olan devreler harici olaraktan bir güç kaynağına ihtiyaç duyarlar. Diğer devreler güç kaynağına ihtiyaç duymazlar. Bununla birlikte assembly komutları yazılmadan yüksek ve orta seviyeli diller kullanılarak da PIC programı yazabiliriz. Örneğin; PICBasic PRO programı ile BASİC temelli bir dilde ve PIC C programı ile C temelli bir dilde PIC programı yazabiliriz. Yüksek ve orta seviyeli diller ile program yazmaya olanak sağlayan yazılımlar genelde ücretlidir. Kısıtlı sürümleri ücretsiz olarak kullanılmaktadır.



Şekil 6.24 PIC programlama adımları

B) PIC Assembly Dili

Assembly dili, bir PIC'e yaptırılması istenen işlerin belirli kurallara göre yazılmış komutlar dizisidir. Assembly dili komutları İngilizce dilindeki bazı kısaltmalardan meydana gelir. Bu kısaltmalar genellikle bir komutun çalışmasını ifade eden cümlenin baş harflerinden oluşur. Böylece komut, bellekte tutulması kolay hale gelir. PIC mikrodenetleyicileri RISC mimarisi ile üretildiklerinden az sayıda komut ile programlanırlar. PIC mikrodenetleyicisi 35 komuta sahip olduğu için programlanması kolaydır. Assembly dilinin temel bileşenleri;

- Etiketler (Labels)
- Komutlar (Instructions)
- İşlenecek veriler (Operands)
- Bildirimler (Direktifler-Directives)
- Yorumlar (Comments)

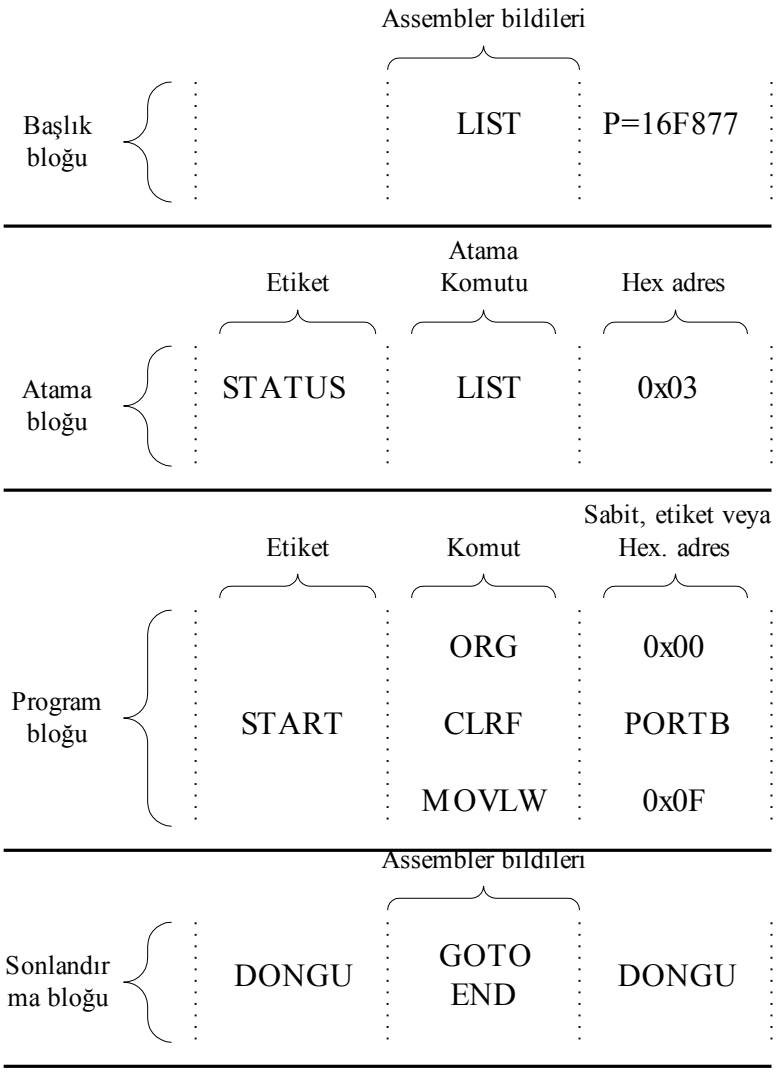
Etiketler PIC'in program ve veri belleğindeki belirli bir adresi isimlerle ifade etmeyi sağlar. Özellikle alt program çağrılmasında kullanılır. Assembler, programı derlerken etiketi gördüğü anda ilgili etiketin adresini otomatik olarak yerine koyup işlemi yapacaktır. Etiketler bir harfle veya "_" karakteri ile başlamalıdır. Program yazılırken Türkçe karakter kullanmamaya dikkat etmeliyiz. Aynı

zamanda etiketler büyük- küçük harf ayrımına karşı duyarlıdır ve en fazla 32 karakter uzunluğundadır. Komut kullanımında dikkate alınması gereken bir noktada yazım hatalarının yapılmamasıdır.

Program kodunun açıklanması için kullanılan yorumlar ";" işaretinden sonra yazılır. Yani baş tarafına ";" konulan satırlar, assembler tarafından hex. kodlara dönüştürülmezler.
Bu satırlar programın geliştirilmesi esnasında hatırlatıcı açıklamaların yazılmasında kullanılır.

Text editörlerinde birbirlerinden farklı uzunlukta girintiler veren TAB özelliği vardır. Bu özellikten yararlanarak assembly komutları üç kolona bölünerek yazılır. Bir assembly programı temel olarak dört bölüme ayrılmaktadır. Bunlar;

- Başlık
- Atama
- Program
- Sonuç bölümleridir.



Şekil 6.25. Assembler komut kolonları

C) PIC Assembly Dilinde Sık Kullanılan İfadeler

 # DEFINE

Birkaç dizini aynı ad altında tutmak için kullanılır. Programı yazmayı kolaylaştıran ifadelerden biridir. Aynı zamanda program kodu içerisinde bir metni başka bir metin ile değiştirir. Define'la başlayan tanımlar program başında yapılmalıdır.

Örnek :

```
#define     AC    1
#define     LED   PORTB,2
```

Bu program satırı ile artık programın herhangi bir yerinde "AC" ifadesi kullanıldığında bu, "1" anlamına gelir. Aynı şekilde program kodunda "LED" ifadesi kullanıldığında bu , PORTB nin 2. ucu anlamına gelir.

#INCLUDE

Programa ek bir dosya ilave edip komutların çalıştırılmasını sağlar.

Örnek :

```
#include <P16F877.inc>
#include " degisken.h"
```

Bu komut satırı, program kodunun daha iyi görünmesi için bazı tanımlamaları her programın başına yazmak yerine bu komutları ayrı bir dosyada yazıp, yeni yazacağımız programımıza eklemeyi sağlar. Örneğin; PIC16F877 için gerekli olan tanımlamaları her programın başına yazmak yerine bir dosyaya atıp, yeni bir program yazarken include tanımı yazılarak bu tanımlar kullanılır. < > ifadesi sistem dosyalarında, " " ifadesi de kullanıcı tarafından yazılan dosyalarda kullanılarak programa eklenir.

CONSTANT

Metin türü olan bir ifadeye sayısal bir değer atamayı sağlar.

Örnek :

```
Constant    MIN=50
Constant    MAX=9600
```

Programımızın derlenmesi yapılırken yukarıda yazılan metin türü ifadelerle karşılaşıldığında bu ifadelere atanan sayısal değerler işleme alınır.

VARIABLE

Metinsel bir ifadeye değiştirilebilir bir sayısal değer atamayı sağlayan ifadedir.

Örnek :

```
Variable    SICAKLIK=25
Variable    DEGER=10
Variable    ORTALAMA=50
```

SET

Önceden tanımlanan bir değişkenin değerini değiştirip yeni bir ifade atamamızı sağlar.

Örnek :

```
SICAKLIK    set     35
DEGER       set     40
```

EQU

Program içerisinde sabit tanımlamamızı sağlayan ifadedir.

Örnek :

```
SAYAC       EQU   0x20
```

Bir etikete veri belleğinin bir adresini atamamızı sağlar.

ORG

Yazılan programımızın, mikrodenetleyicinin veri belleğinin hangi adresinden başlanıp yükleneceğini belirler.

Örnek :

```
ORG   0x000
```

IF, ELSE, ENDIF

IF şart deyimidir. Belirlenen koşul sağlandığında IF deyimini takip eden program kodları işleme koyulur. Eğer şart sağlanmıyorsa ELSE deyiminden sonraki komut satırları işleme girer. Eğer IF bildiriminden sonra ELSE kullanılmazsa ENDIF bildiriminden sonra gelen komutlar işleme koyulur. Kullanılan IF bildiriminden sonra mutlaka ENDIF bildirimide kullanılmalıdır.

Örnek :

```
IF MAX==1000
        movlw   h ‘01’
else
        movlw   h ‘02’
endif
```

Bu ifadeye göre MAX değeri 1000’e eşitse movlw h ‘01’ komutu, değilse movlw h ‘02’ komutu yüklenir.

END

Program sonunu belirten ifadedir.

Örnek :

```
……
End
```

WHILE, ENDW

While ifadesinden sonra yazılan koşul sağlanıncaya kadar WHILE ile ENDW ifadeleri arasındaki program kodları derlenir.

Örnek :

```
While SAY<15
    SAY=SAY+2
ENDW
```

Bu komut satırına göre SAY değişkeninin değeri 15'den küçük olduğu sürece 2 arttırılır.

IFDEF

#Define ile atanan metnin tanımlı olduğu durumda *ifdef* bildirimine kadar olan program kodunun derlenmesini sağlar.

Örnek :

```
#define led 1
……
……
ifdef  led
```

_CONFIG

Bildirimi takip eden açıklamaları işlemcinin konfigürasyon bitlerine aktarmayı sağlayan ifadedir.

Örnek :

```
_CONFIG h 'FFFF'
```

D) PIC Komut Seti

PIC16F877 nin 14-bit sözcük uzunluğuna sahip toplam 35 komut vardır. Komutlar 3 grupta incelenir. Komutların çoğu, bir saat çevrimlik sürede uygulanır. Bir saat çevrim süresi; osilatörün frekansının ¼ ‘ü kadar olan bir süreye eşittir. Call, goto gibi bazı komutlar 2 saat çevrimi sürede işlenir.

Komut Formatları

Bazı komutlar çalışırken, komutun ve işlemesi sırasında oluşan durumlara bağlı olarak, STATUS kayıtçısının gerekli olan bitleri değişir.

- **BYTE Yönlendirmesi Yapan Yazmaç İşlemleri:**

d=0 için hedef W, d=1 için hedef f, f=7 bit yazmaç adresi

13 8	7	6 0
İşlem Kodu	D	f (yazmaç)

- **Bit Yönlendirmesi Yapan Yazmaç İşlemleri:**

b=3 bit adres f=7 bit yazmaç adresi

13 10	9 7	6 0
İşlem Kodu	b (bit no)	f (yazmaç)

- **Denetim ve Sabit/Sayısal (literal) İşlemler:**

13 8	7 0
İşlem Kodu	k (literal)

k= 8 bit hazır (immediate) değer

Yalnız call ve goto komutlarında kullanılan biçim:

13 11	10 0
İşlem Kodu	k (literal)

k=11 bit hazır (immediate) değer

Komutların tümü, işlem biçimi tablolarında da gösterildiği şekilde; bit yönlendirmeli, byte yönlendirmeli ve son olarak da, literal ve kontrol komutları olarak üç bölümde sınıflandırılır.

PIC16F877 Komut Kümesi

PIC16F877 komut tablosu ve komut tablosunda kullanılan semboller, aşağıdaki komut tablosunda yer almaktadır.

Komut Yazılımı	Komut Tanımlaması	Çevrim Süresi	14-bitlik Opcode MSb LSb	STATUS Etkisi
BYTE Yönlendirmeli Komutlar				
ADDWF f, d	W ile f 'yi topla	1	00 0111 d f f f f f f f	C,DC,Z
ANDWF f, d	W ile f 'yi AND 'le	1	00 0101 d f f f f f f f	Z
CLRF f	f 'yi sil	1	00 0001 l f f f f f f f	Z
CLRW --	W 'yı sil	1	00 0001 0xxx xxxx	Z
COMF f, d	f 'nin tersini al	1	00 1001 d f f f f f f f	Z
DECF f, d	f 'yi bir azalt	1	00 0011 d f f f f f f f	Z
DECFSZ f, d	f 'yi bir azalt, f = 0 ise bir komut atla	1 (2)	00 1011 d f f f f f f f	
INCF f, d	f 'yi bir arttır	1	00 1010 d f f f f f f f	Z
INCFSZ f, d	f 'yi bir arttır, f = 0 ise bir komut atla	1 (2)	00 1111 d f f f f f f f	
IORWF f, d	W ile f 'yi XOR 'la	1	00 0100 d f f f f f f f	Z
MOVF f, d	f 'yi taşı	1	00 1000 d f f f f f f f	Z
MOVWF f	W 'yı f 'ye taşı (W → f)	1	00 0000 l f f f f f f f	
NOP --	İşlem yapma	1	00 0000 0xx0 0000	
RLF f, d	f 'yi birer bit sola döndür	1	00 1101 d f f f f f f f	C
RRF f, d	f 'yi birer bit sağa döndür	1	00 1100 d f f f f f f f	C
SUBWF f, d	f 'den W 'yı çıkart	1	00 0010 d f f f f f f f	C,DC,Z
SWAPF f, d	f 'nin dörtlü bitlerinin yerini değiştir	1	00 1110 d f f f f f f f	
XORWF f, d	W ile f 'yi XOR 'la	1	00 0110 d f f f f f f f	Z
BIT Yönlendirmeli Komutlar				
BCF f, b	f 'nin b. bitini sil	1	01 00bb b f f f f f f f	
BSF f, b	f 'nin b. bitini bir yap	1	01 01bb b f f f f f f f	
BTFSC f, b	f 'nin b. biti "0" ise bir komut atla	1 (2)	01 10bb b f f f f f f f	
BTFSS f, b	f 'nin b. biti "1" ise bir komut atla	1 (2)	01 11bb b f f f f f f f	
Literal ve Kontrol Komutları				
ADDLW k	k 'yı W 'ya ekle	1	11 111x kkkk kkkk	C,DC,Z
ANDLW k	k 'yı W ile AND 'le	1	11 1001 kkkk kkkk	Z
CALL k	k alt programını çağır	2	10 0kkk kkkk kkkk	
CLRWDT --	WDT yi sil	1	00 0000 0110 0100	$\overline{TO}$, $\overline{PD}$
GOTO k	k adresine git	2	10 1kkk kkkk kkkk	
IORLW k	k ile W 'yı OR 'la	1	11 1000 kkkk kkkk	Z
MOVLW k	k 'yı W 'ya taşı	1	11 00xx kkkk kkkk	
RETFIE --	Kesmeden geri dön	2	00 0000 0000 1001	
RETLW k	k 'yı W 'ya yükle ve geri dön	2	11 01xx kkkk kkkk	
RETURN --	Alt programdan geri dön	2	00 0000 0000 1000	
SLEEP --	Uyku moduna geç	1	00 0000 0110 0011	$\overline{TO}$, $\overline{PD}$
SUBLW k	W 'yı k 'dan çıkart	1	11 110x kkkk kkkk	C,DC,Z
XORLW k	k ile W 'yı XOR 'la	1	11 1010 kkkk kkkk	Z

Sembol Tanımlamaları :

f → Register File Adress: kayıtçı adı veya adresi (0x00 ile 0x7F)
w → Akümülatör, çalışma kayıtçısı
b → Bit tanımlayıcısı; 8 bitlik kayıtçının 0~7 arasındaki bir biti veya etiket (EQU komutu ile adresi tanımlanmış olması gerekir)
d → Destination : Gönderilecek yer; komutun çalıştırılmasından sonra sonucun nereye yazılacağını belirler. d = 0 → W kayıtçısına, d = 1 → dosya kayıtçısına
k → Sabit bir sayı (0x0C veya $0C_H$, 00001100_B, 10_D) veya adres etiketi
x → "0" yada "1" önemli değil
TO → Zaman aşımı biti (Time-out bit)
PD → Güç kesimi biti (Power-down)

NOT:

- Eğer kaynak işlenen I/O portu ise mikrodenetleyici pinlerindeki durum okunur.
- Eğer bu komut TMR yazmacı üzerinde uygulanırsa ve d=1 ise bu zamanlayıcıya atanan önölçekleyici otomatik olarak 0 olur.
- Eğer PC modifiye edilmişse veya test sonucu 1 ise komut iki saat çevriminde işlenir.
- C (Carry) DC (Digital Carry) 2, TO (Time Out) PD (Power Down) yapılan işlemler sonucu etkilenen bayraklardır. Bu bayraklar birçok uygulamada kontrol edilerek sistemin çalışması değiştirilebilir.

Kayıtçı Adresleme Modları

PIC lerde başlıca üç tip adresleme modu vardır:

1. **Anında(Immediate) Adresleme:**

Örnek : MOVLW H'0F' ; W = OF → Komut Sözcüğü : 11 0000 0000 1111 =300F

2. **Doğrudan (Direct) Adresleme :** 14 bitlik komut sözcüğünün 7 biti kayıtçı adresini tanımlar. 8. ve 9. bitler STATUS un RP0 ve RP1 bitlerinden elde edilir.

Örnek :

```
Z    EQU    d'2' // Status kayıtçısının 2. biti Z (zero) dur.
BTFSS STATUS, Z → Komut Söz. : 01 1101 0000 0011=1D03
```

3. **Dolaylı (Indirect) Adresleme :** 8 bitlik kayıtçı adresi FSR (özel fonksiyonlu kayıtçı) kayıtçısına yazılır. FSR nin işaret ettiği adresin içeriği için INDF kullanılır.

```
INDF = [FSR];   okuma
[FSR] = INDF;   yazma
```

Örnek: h'20' – h '2F' RAM bölgesini temizleyen (sıfırlayan) bir program.

```
Temizle;          20H-2F H arasını temizler.
                  movlw  0x20;          Göstergeye başlangıç değerini (adresi) ver .
                  movwf FSR;            RAM'a git
Sonraki:
                  clrf  INDF;           INDF yazmacını temizle
                  incf FSR, F;          Göstergeyi bir arttır.   (d = 1)
                  btfss FSR, 4;         Hepsi yapıldı mı?
                  goto Sonraki;         Temizlenecek alan bitmedi, sonrakine git.
                  Başka_Kısıma_Geç;     Temizlenecek alan bitti.
```

PIC Assemblyde Sayıların ve Karakterlerin Yazımı

Heksadesimal Sayılar

Heksadesimal sayılar "0x", "0" veya "h" harfleriyle başlamalıdır. Örneğin, STATUS kayıtçısına 03 adresi atamak için ;

```
STATUS    EQU     0x03
          EQU     3
          EQU     03
          EQU     03h
          EQU     h '03'
```

kullanılır. MOVLW komutu kullanılarak w kayıtçısı içerisine yüklenecek olan FF heksadesimal sabitler ise ;

```
MOVLW     0xFF
          h 'FF'
```

Binary Sayılar

Binary sayılar yazılırken b harfi ile başlamalıdır. Örneğin 00001010 binary sayısı W kayıtçısının içerisine yüklenirken aşağıdaki gibi yazılmalıdır.

```
MOVLW     b '00001010'
```

Desimal Sayılar

Desimal sayıların başına d harfi koyularak tırnak içerisinde yazılır. Örneğin 15 desimal sayısının W kayıtçısının içerisine yüklerken aşağıdaki gibi yazılmalıdır.

```
MOVLW     d '15'
```

ASCII Karakterler

ASCII karakterler yazılırken RETLW komutu ile yazılır.

```
RETLW     'A'
RETLW     'T'
```

F) Komutların Kullanışı

Komut seti tabloları, assembler derleyicisinin kabul ettiği komut sözcükleri ve bunların söz dizimi kuralları hakkında bilgi sağlamaktadır. Bu bölümde ise her komut, kendisini oluşturan temel elemanlarıyla birlikte incelenip, örneklendirilmektedir. Ayrıca kullanılabilecek her bir komut için; uyguladığı işlem (operation) ve neyi yada neleri işlediği; yani işleçler (operands) hakkında bilgi verilmektedir. İşlem sonucunda, STATUS yazmacı da etkileniyorsa belirtilmiştir. Komutun yazıldığı satırdaki, köşeli parantez içinde belirtilmiş olan etiket adı yazılabilir, ancak yazılması zorunlu değildir. İşlev satırlarında parantez içine alınan kayıtçı örneğin (W) gibi, o yazmacın

içeriğini göstermektedir. Bir yazmacın belirli bitlerinin değerleri, (kayıtçı_adı<…>) ile gösterilmiştir. Her komut satırındaki etiket birinci, komut ikinci, komutun kullandığı işleçler ise üçüncü bloka yazılmaktadır.

1-) ADDLW Bir sayı/sabit ile W nin içeriğini topla

Söz dizim kuralı : [etiket] ADDLW k

işleçler : $0 \leq k \leq 255$

işlevi : (W) + k → (W)

Status etkisi : C, DC, Z

Tanımı : W'nin içeriğini 8 bitlik k literali ile toplar ve sonucu W' ye aktarır.

Örnek : ADDLW h'FF'

Komuttan önce k=h'FF' ve w=h'01' ise, komut çalıştıktan sonra W = 00h olur. Toplam sonucu, FFh' tan büyük olduğu zaman, elde biti W yazmacına sığmaz. Elde biti, STATUS yazmacının içinde C-Carry bitinde (STATUS, 0) tutulur. W yazmacının içeriği (değeri) sıfırsa, status yazmacının zero biti de 1(set-zero-flag) yapılır. Yani Z = 1; C =1 olur.

2-) ADDWF Bir yazmaç içeriği (f) ile W nin içeriğini topla

Söz dizim kuralı : [etiket] ADDWF f,d

işleçler : $0 \leq f \leq 127$ ve $d \in (0,1)$

işlevi : (W) + f → (hedef)

Status etkisi : C, DC, Z

Tanımı : W'nin içeriğini, 7 bitlik adresi olan f kayıtçısının içeriği ile toplar ve sonucu d = 0 ise W'ye d=1 ise f ye aktarır.

Örnek : ADDWF f,0

Bu komuttan önce W=h'10', f=h'10' ise komuttan sonra W=h'20' ve f=h'10' olur.

Örnek : ADDWF f,1

Bu komuttan önce W=h'10', f=h'10' ise komuttan sonra W=h'10' ve f=h'20' olur. Toplama sonucu h'FF' değerini aşarsa, Status yazmacı, aynı literalle toplama komutundaki gibi etkilenir.

3-) ANDLW Bir sayı ile W nin içeriğine AND İşlemini Uygula

Söz dizim kuralı : [etiket] ANDLW k

işleçler : $0 \leq k \leq 255$

işlevi : W AND k → (W)

Status etkisi : Z

Tanımı : W'nin içeriğini k ile AND 'le, sonucu W ye aktar.

Örnek : ANDLW h'03'

Bu komuttan önce W=h'01' ise, Komut .VE. işlemini uygular. Komut sonucu W=h'01' olur.

4-) ANDWF Yazmaç içeriğini W nin içeriği ile AND' le

Söz dizim kuralı : [etiket] ANDWF f,d

İşleçler : $0 \leq f \leq 127$ ve $d \in (0,1)$

İşlevi : W AND f → (hedef)

Status etkisi : Z

Tanımı : W'nin içeriğim f yazmacının içeriği ile AND 'le ve sonucu d=0 ise W 'ye yükle, d=1 ise f ye yükle.

Örnek : ANDWF f,0

Bu komut çalışmadan önce W=h'03', f=h'07' ise işlem yaptıktan sonra W=h'03', f=h'07' olur.

Örnek : ANDWF f,1

Komut çalışmadan önce W=h'03', f=h'07' ise işlem yaptıktan sonra W=h'03', f=h'03' olur.

STATUS 'ün etkilenmesi : AND işlemi sonucu, h'00' olsaydı, STATUS registerin 2. biti olan Z biti 1(set) yapılırdı.

MASKELEME özelliği : AND mantıksal işleminin maskeleme özelliği vardır. Mantıksal durumunun değişmesini istemediğimiz bitleri, 1 ile AND 'lersek, diğer bitler 0 olurken maskelediğimiz bitler değişmez.

5-) BCF Yazmacın belirlenen bitini sıfırla (clear)

Söz dizim kuralı : [etiket] BCF f,b

İşleçler : $0 \leq f \leq 127$ ve $0 \leq b \leq 7$

işlevi : $0 \rightarrow f(b)$

Status etkisi : Yok

Tanımı : f yazmacının b. bitini 0 yap.

Örnek : BCF PORTD,0

Komutu çalışınca PORTD yazmacının ilk biti 0 yapılır. PORTD 'nin 0. bitine bağlı bir led yanıyorsa, bu komutla söndürülür.

6-) BSF Yazmacın belirlenen bitini bir (set) yap

Söz dizim kuralı : [etiket] BSF f,b

İşleçler : $0 \leq f \leq 127$ ve $0 \leq b \leq 7$

işlevi : $1 \rightarrow f(b)$

Status etkisi : Yok

Tanımı : f yazmacının b. bitini 1(set) yap.

Örnek : BSF PORTD,0

Komutu çalışınca PORTD yazmacının ilk biti 1 yapılır. PORTD'nin 0. bitine bağlı bir led yanmıyorsa, bu komutla yakılabilir. BCF ile BSF komutları ardı ardına kullanılarak, bir kare dalga sinyali elde edilir.

7-) BTFSC Yazmacının belirlenen biti 0 ise, bundan sonraki komutu atla

Söz dizim kuralı : [etiket] BTFSC f,b

İşleçler : $0 \leq f \leq 127$ ve $0 \leq b \leq 7$

İşlevi : $0 \rightarrow f(b)$

Status etkisi : Yok

Tanımı : Yazmacının b. bitinin 0 olup olmadığı kontrol edilir. Eğer sıfır ise bu komutun altındaki komut işlenmez, bir sonraki komuta sapılır. Aksi durumda ise sıradaki komut uygulanır.

Örnek :

```
Basla
        BTFSC PORTB,0     ; komut çalışınca PORTB yazmacının ilk bitinin
                          ; 0 olup-olmadığı kontrol edilir. Eğer sıfır ise bu
                          ; komutun hemen altındaki komut işlenmez, bir
                          ; sonraki komuta sapılır. Aksi durumda ise
                          ; sıradaki komut uygulanır.
        GOTO Basla        ; PORTB'nin 0. biti 0 değilse işlenecek, tekrar
                          ; başa dönecek.
        BSF  PORTB,1      ; PORTB'nin 0. biti 0 olunca işlenecek, aynı bit
                          ; bu komutla 1 yapılacak. Eğer pinde led varsa
                          ; yanacak. Böylece pinde bir kare dalga sinyali
                          ; oluşur.
```

8-) BTFSS Yazmacın belirlenen biti 1 ise, bundan sonraki komutu atla

Söz dizim kuralı : [etiket] BTFSS f,b

İşleçler : $0 \leq f \leq 127$ ve $0 \leq b \leq 7$

işlevi : 1 → f(b)

Status etkisi : Yok

Tanımı : Yazmacının b. bitinin 1 olup olmadığı kontrol edilir. Eğer bir ise bu komutun altındaki komut işlenmez, bir sonraki komuta sapılır. Aksi durumda ise sıradaki komut uygulanır.

Örnek .a.

```
Basla
      BTFSS PORTA,0     ; komut çalışınca PORTA yazmacının ilk bitinin
                        ; 1 olup-olmadığı kontrol edilir. Eğer bir ise bu
                        ; komutun hemen altındaki komut işlenmez, bir
                        ; sonraki komuta sapılır. Aksi durumda ise
                        ;sıradaki komut uygulanır. Porta'nın ilk bitine bir
                        ; buton bağlı olsun. Butona basılıp, basılmadığı
                        ; bu komutla kontrol edilebilir.
      GOTO Basla        ; PORTA'nin 0. biti 1 değilse işlenecek, tekrar
                        ; başa dönecek.
      BSF   PORTB,1     ; PORTA'nin 0. biti 1 ise işlenecek, bu komutla
                        ; PortB'nin 2. bitine bağlı led yanar.
```

Örnek .b.

```
      BTFSC STATUS,0    ; Bu komutla, işlem sonucunun h'FF' sayısından
                        ; büyük olup olmadığım denetleyebilir. Status
                        ; yazmacının 0. biti (C) 0 ise bir komut atla
                        ; anlamına gelen bu komut sık kullanılır.
```

Örnek .c.

```
      BTFSC STATUS,2    ; Bu komut ise, toplama işleminin sonucunun 1
                        ; olup-olmadığını, status yazmacının 2. bitinin 1
                        ; olup olmadığına bakarak denetler. Yazmacın
                        ; 2. biti (Z) 1 ise bir komut atlar.
```

9-) CALL Altprogramı çağır

Söz dizim kuralı : [etiket] CALL k

işleçler : $0 \leq k \leq 2047$

işlevi : (PC) + 1 → TOS,

TOS: Yığının üstü (Top of Stack), k → (PC<10:0>), PCLATH<4:3> →PC<12:11>

Status etkisi : Yok

Tanımı : Altprogramı çağırır. Önce PC' yi bir arttırır ve yığının üstüne koyar. Sonra altprogram adresi PC' nin <10:0> bitlerine yüklenir. PCLATH' in <4:3> bitlerindeki değerler, PC' nin üst bitleri olan <12:11> arasındaki bitlere yüklenir. CALL işlemi iki saat çevrimi sürede uygulanan bir dallanma komutudur.

Örnek : REF1 CALL Gönder

Bu komuta gelindiğinde PC bir arttırılarak TOS 'a konur. TOS 'da REF1 'in adresi var. Böylece, TOS 'da komut uygulandıktan sonra dönülecek adres oluşturulmuş olur. Bundan sonraki aşamada PC'ye Gönder altprogramının adresi oluşturulur, yani altprograma sapılır. Alt program komutları sırası geldikçe uygulanacak altprogramı sonlandıran RETURN ile birlikte, TOS 'daki değer PC' ye geri yüklenecektir ki bundan sonraki komut uygulanabilsin. Bir CALL komutuyla sapılan komut takımının bulunduğu adresten dönüş için, mutlaka altprogramın sonlandırıcısı RETURN gerekir.RETURN ileride de anlatılacak. RETURN uygulandığında TOS deki adres PC'ye yüklenir. PCLATH<4:3> bitleri ise bellek sayfalarının değerini içerdiği için üst bitlere yüklenerek altprogramın bulunduğu doğru adrese sapılması sağlanır.

10-) CLRF Yazmaç İçeriğini sil

Söz dizim kuralı : [etiket] CLRF f

işleçler : $0 \leq f \leq 127$

işlevi : h'00' → (f) ve 1 → Z

Status etkisi : Z

Tanımı : f yazmacının içeriği sıfırlanır ve değeri sıfır olduğu için status yazmacının zero biti 1 (set) yapılır.

Örnek : CLRF TRISD

D Portunun yönlendiricisi olan TRISD yazmacının tüm bitleri sıfır yapılmıştır. Böylece D Portu çıkış olarak belirlenmiştir. Bu portta ledler, veya LCD, 7SD ...vb. birimler olabilir. TRISD 'nin sıfırlanması sonucu, status yazmacı zero biti de set edilmiştir.

11-) CLRW W yazmacının içeriğini sil

Söz dizim kuralı : [etiket] CLRW

işleçler : Yok

işlevi : h'00' → (W) ve 1 →Z

Status etkisi : Z

Tanımı : W yazmacının içeriği sıfırlanır ve değeri sıfır olduğu için status yazmacının zero biti 1 (set) yapılır.

Örnek : CLRW ; W yazmacı temizlendi. Status Z biti 1 oldu.

12-) CLRWDT WDT zamanlayıcı içeriğini sil

Söz dizim kuralı : [etiket] CLRWDT

İşleçler : Yok

işlevi : h'00' → WDT

0→ WDT önbölücü sabit(prescaler), 1 → TO', 1 → PD'

Status etkisi : TO, PD

Örnek : CLRWDT

Komut uygulanmadan önce WDT'nin içeriği ne olursa olsun, komut çalıştırıldıktan sonra WDT sayacı ve önbölücüsü 0 'lanır. Aynı zamanda TO' ve PD' bitleri 1 olur.

13-) COMF Yazmaç içeriğinin tersini al (f'nin tümleyeni)

Söz dizim kuralı : [etiket] COMF f,d

İşleçler : $0 \leq f \leq 127$ $d \in (0.1)$

işlevi : (f)' → (hedef)

Status etkisi : Z

Tanımı : f yazmacının içeriği terslenir ve d değeri sıfır ise sonuç W yazmacına, bir ise f yazmacına yüklenir.

Örnek : COMF f,0

Bu komuttan önce W=h'02' ve f=h'01'ise, komuttan sonra W=h'FE' ve f=h'01' olur. Sonuç ters alma işleminde sıfır olmuş ise Z bit de 1 yapılır.

Örnek : COMF f,1

Bu komuttan önce W=h'02' ve f=h'01 'ise, komuttan sonra W=h'02' ve f=h'FE' olur. Z bit sıfırlanmaz sonuç 0 'dan farklı.

14-) DECF Yazmaç içeriğini bir azalt

Söz dizim kuralı : [etiket] DECF f,d

İşleçler : $0 \leq f \leq 127$ ve $d \in (0,1)$

İşlevi : (f) -1 → (hedef)

Status etkisi : Z

Tanımı : f yazmacının içeriği bir azaltılır ve d değeri sıfır ise sonuç W yazmacına, bir ise f yazmacına yüklenir.

Örnek : DECF SAYAC,0

Sayacın içindeki değer her ne ise, bir azaltılır ve sonuç, d' nin 0 olması durumunda W 'ye, aksi halde ise SAYAÇ yazmacına yüklenir. Sonuç 0 ise status'ün Z biti 1 yapılır.

15-) DECFSZ Yazmaç içeriğini bir azalt, 0 ise bir komut atla

Söz dizim kuralı : [etiket] DECFSZ f,d

İşleçler : $0 \leq f \leq 127$ ve $d \in (0,1)$

İşlevi : (f)-1 → (hedef) ve Sonuç = 0 ise atla

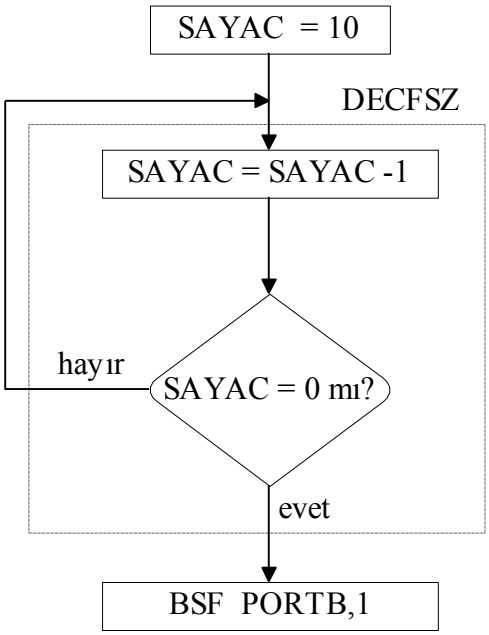
Status etkisi : Yok

Tanımı : f yazmacının içeriği 1 azaltılır ve sonuçta oluşan değer sıfır ise, bu komutu izleyen komut atlanır. Sonuçta d=0 ise W ' ye 1 ise f' ye yüklenir. Komut atlamayla sonuçlanırsa, ikinci çevrim süresinde NOP uygulayarak, toplam iki saat çevrim süresinde işlenir. Atlama olmadığı durumda uygulanması bir saat çevrimi süredir.

Örnek :

```
Azalt   DECFSZ    SAYAC,1   ; SAYAÇ 1 azaltılır, sonuç 0 ise GOTO komutu atlanır.
        GOTO  Azalt         ; Sonuç 0 değilse Azalt etiketine sapılır.
        BSF  PORTB,1        ; Sonuç 0 ise PORTB'nin RB1=1 edilir.
```

SAYAC' in başlangıç değeri 10 ise yukarıdaki azalt döngüsü, 10 kez tekrar edilir. 10. tekrarda SAYAÇ değeri sıfırlanmıştır. Goto sapma komutundan sonraki komutla devam edilir.



16-) GOTO Adrese git

Söz dizim kuralı : [etiket] GOTO k

İşleçler : $0 \leq k \leq 2047$

İşlevi : k → PC<10:0>

PCLATH<4:3> → PC<12:11>

Status etkisi : Yok

Tanımı : GOTO koşulsuz bir sapma komutudur, k' nin adresi neyse PC'ye <10:0> bitlerine yüklenir. Belek sayfası neyse PCLATH<4:3> bitleri PC'nin üst bitlerine yüklenir ve adrese sapılır. Bu komut, iki saat çevrimi sürede uygulanır.

Örnek : Basa_Tası GOTO Bas

Komuttan önce PC' da Basa_Tası etiketinin adresi vardır. Komut çalıştırıldıktan sonra ise PC 'de Bas etiketinin adresi oluşur.

17-) INCF Yazmaç içeriğini bir arttır

Söz dizim kuralı : [etiket] INCF f,d

İşleçler : $0 \leq f \leq 127$ ve $d \in (0,1)$

işlevi : (f) + 1 → (hedef)

Status etkisi : Z

Tanımı : f yazmacının içeriği bir arttırılır ve d değeri sıfır ise sonuç W yazmacına, bir ise f yazmacına yüklenir.

Örnek : INCF SAYAC,0 → W = SAYAC + 1;

Sayacın içindeki değer her ne ise bir arttırılır ve sonuç d'nin 0 olması durumunda W ye, aksi halde ise f yazmacına yüklenir.

18-) INCFSZ Yazmaç içeriğini bir arttır, 0 ise bir komut atla

Söz dizim kuralı : [etiket] INCFSZ f,d

İşleçler : $0 \leq f \leq 127$ ve $d \in (0,1)$

işlevi : (f)+1 → (hedef)
sonuç=0 ise atla

Status etkisi : Yok

Tanımı : f yazmacının içeriği bir arttırılır ve sonuçta oluşan değer sıfır ise, bu komutu izleyen komut atlanır. Sonuç, d=0 ise W 'ye, 1 ise f 'ye yüklenir. Komut atlamayla sonuçlanırsa, ikinci çevrim süresinde NOP uygulayarak, toplam iki saat çevrim süresinde işlenir. Atlama olmadığı durumda uygulanması bir saat çevrimi süredir.

Örnek :

```
Art   INCFSZ  SAYAÇ, 1        ; SAYAÇ 1 artar, sonuç 0 ise BCF komutu atlanır.
      GOTO    Art             ; Sonuç 0 değil ise Art' a gidilir.
      BCF     PORTB,1         ; Sonuç 0 ise PORTB 'nin 1. biti 0 edilir.
```

19-) IORLW Bir sayı ile W'nin içeriğine OR işlemini uygula

Söz dizim kuralı : [etiket] IORLW k

İşlevi : (W) OR k → (W)

Status etkisi : Z

Tanımı : W yazmacının içeriği k literali ile OR' lanır. Sonuç W 'ya yüklenir. Mantıksal işlem sonunda oluşan değer sıfır ise, status Z biti 1 yapılır.

Örnek : IORLW h'0F'

Komut öncesi W=h'F0' ise, komut sonrası W=h'FF'olur.

20-) IORWF Yazmaç içeriği ile W nin içeriğine OR işlemini uygula

Söz dizim kuralı : [etiket] IORWF f,d

İşleçler : $0 \leq f \leq 127$ ve $d \in (0.1)$

İşlevi : (W) OR f → (hedef)

Status etkisi : Z

Tanımı : W yazmacının içeriği k literali ile OR' lanır ve sonuç d=0 ise W 'ya, d=1 ise f 'ye yüklenir.Mantıksal işlem sonunda oluşan değer sıfır ise, status Z biti 1 yapılır.

Örnek : IORWF f,0

Komuttan önce W=h'10' ve f=h'01' ise, komuttan sonra W= h'11', f=h'01 'olur.

Örnek : IORWF f,1

Komuttan önce W=h'10" ve f=h'01' ise,komuttan sonra W=h'10',f=h'11'olur.

21-) MOVLW W ya bir sayı/sabit yükle

Söz dizim kuralı : [etiket] MOVLW k

İşleçler : $0 \leq k \leq 255$

İşlevi : k → (W)

Status etkisi : Yok

Tanımı : W yazmacının içeriği k olur.

Örnek : MOVLW k

Komut öncesi W nin değeri ne otursa olsun, komuttan sonra k literalinin değeri ile yüklenir.

22-) MOVF Yazmaç içeriğini hedefe tası (f'yi yükle)

Söz dizim kuralı : [etiket] MOVF f,d

İşleçler : $0 \leq f \leq 127$ ve $d \in (0,1)$

İşlevi : (f) → (Hedef)

Status etkisi : Z

Tanımı : f yazmacının içeriği; d 0 ise W yazmacına, 1 ise kendisine yüklenir, özellikle d=1 olduğu durumda, f nin içeriğinin 0 olup olmadığına bakılması yararlı olur. Çünkü bu durumda status'ün Z biti değişir.

Örnek : MOVF f,0

Komuttan önce W=h'0F ve f=h'01' otsun, komuttan sonra W=h'01'vef=h'01'olur.

Örnek : MOVF f,1

Komuttan önce W=h'0F ve f=h'01'olsun, komuttan sonra W=h'0F'vef=h'01'olur.

23-) MOVWF W nin içeriğini f yazmacına taşı

Söz dizim kuralı : [etiket] MOVWF f

İşleçler : $0 \leq f \leq 127$

İşlevi : (W) → (f)

Status etkisi : Yok

Tanımı : W yazmacının içeriği f yazmacına taşınır.

Örnek : MOVWF SAYAC

Komutu uygulanmadan önce, SAYAÇ yazmacının içeriği ne olursa olsun komut uygulandıktan sonra W yazmacının içeriği SAYAÇ' a yüklenir.

24-) NOP İşlem yok

Söz dizim kuralı : [etiket] NOP

İşleçler : Yok

işlevi : Yok

Status etkisi : Yok

Tanımı : Hiçbir şey yapılmadan bir saat çevrimi süre alır.

Örnek : NOP ;Hiçbir işlem yapılmadan, bir çevrimlik süre geçirir.

25-) RETFIE Kesme altprogramından geri dön

Söz dizim kuralı : [etiket] RETFIE

İşleçler : Yok

işlevi : TOS → PC ve 1 →GIE

Status etkisi : Yok

Tanımı : Kesme altprogramından, dönmek için kullanılır. Dönüş hazırlığından başka bir işlem yapmaz. Dönüş yapılacak adres TOS 'de olduğu için, TOS değeri PC' ye yüklenir. INTCON kesme yazmacının, GIE biti set edilir. Komut iki saat çevriminde işlenir.

Örnek : RETFIE ; Bu komut uygulanınca PC=TOS ve GIE=1 olur.

26-) RETLW Altprogramından W'ye bir sayı/sabit yükle ve geri dön

Söz dizim kuralı : [etiket] RETLW k

İşleçler : $0 \leq k \leq 255$

İşlevi : k → W ve TOS → PC

Status etkisi : Yok

Tanımı : Altprogramdan, W ' ye k literali yüklenmiş olarak dönmek için kullanılır. Dönüş hazırlığı TOS değerinin PC' ye aktarılmasıyla yapılır. Komut iki saat çevrimi sürede işlenir.

|

Örnek : RETLW h'21'

Bu komut uygulandıktan sonra W yazmacına h'21' yüklenir. PC' ye TOS değeri yerleştirilir. Özellikle altprogramdan değerler dizisinden biri ile dönmesi istendiğinde kullanılır.

```
TABLO   ADDWF SAYAC     ;SAYAC 'ın aldığı değer kaç ise, o RETLW ye sapar.
        RETLW 21h       ;W önceden h'01 'se komuttan sonra W=h'21' le
        RETLW 22h       ;W önceden h'02'se komuttan sonra W=h'22' le
        RETLW 23h       ;W önceden h'03'se komuttan sonra W=h'23' le
        RETLW 24h       ;W önceden h'04'se komuttan sonra W=h'24' le
```

döner. W' nin tablo kesimine sapmadan önceki değerine göre işlem görür.

27-) RETURN Altprogramdan geri dön

Söz dizim kuralı : [etiket] RETURN

İşleçler : Yok

İşlevi : TOS→PC

Status etkisi : Yok

Tanımı : Altprogramından TOS 'daki adresle geri döner. Komut iki saat çevrimi sürede işlenir.

Örnek : RETURN

Bu komut uygulandıktan sonra PC' ye TOS değeri yerleştirilir.

28-) RLF Yazmaç bitlerini sola doğru döndür

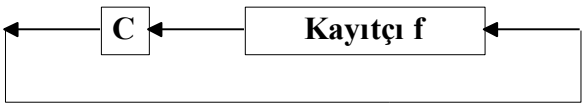
Söz dizim kuralı : [etiket] RLF f,d

İşleçler : $0 \leq f \leq 127$ ve $d \in (0,1)$

İşlevi : Tanım kısmında ayrıntılandırılmıştır.

Status etkisi : C

Tanımı: f yazmacındaki bitleri bir bit sola doğru yerleştir. Böylece 0. bitin değeri 1. bite, 1. bitin değeri 2. bite, ...,6. bitin değeri 7. bite yerleşir. Yazmaç 8 bitlik olduğundan 7. bitin değeri status yazmacının C bitine yerleştirilir. Daha sonra C bitteki değer, f yazmacının 0. bitine aktarılır. Böylece hiçbir bit bozulmadan sola doğru kaymış olur. d=0 ise, sonucu W ye, aksi durumda d=1 f ye taşır. C biti f yazmacının en üst bilinin değerini taşır.



Örnek :

RLF SOL,1 ;komuttan önce SOL=H'01', ve C=1 ise, komut çalışınca SOL=b'0000 0011' = 03h ve C=0 olur.

RLF SOL,1 ;komut bir kez daha çalışınca, SOL=b'0000 0110' ve C=0 olur.

29-) RRF Yazmaç bitlerini birer bit sağa aktar

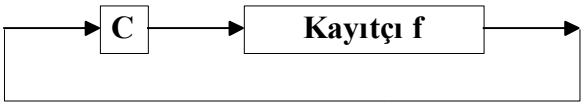
Söz dizim kuralı : [etiket] RRF f,d

İşleçler : $0 \leq f \leq 127$ ve $d \in (0,1)$

İşlevi : Tanım kısmında ayrıntılandırılmıştır.

Status etkisi : C

Tanımı : f yazmacındaki bitleri bir bit sağa doğru yerleştirir.



Örnek :

RRF SAG,1 ; komuttan önce SAG=H'02', ve C=0 ise, komut çalışınca
; SAG=b '0000 0001'=01h ve C=0 olur.

RRF SAG,1 ; komut bir kez daha çalışınca, SAG=b '0000 0000' ve C=1 olur.

30-) SLEEP Standby (uyku) moduna gir

Söz dizim kuralı : [etiket] SLEEP

işleçler : Yok

işlevi : h'00' → WDT,

0 → WDT ön bölücü sabiti (prescaler), 1→TO' ve 0→PD'

Status etkisi : TO,PD

Tanımı : PD , güç kesim biti temizlenir. TO , süre aşımı biti 1 olur.WDT ve ön bölücü Sabit de sıfırlanır. Osilatörün de durmasıyla, işlemci uyuma oduna geçer. PIC bu durumda çok az güç harcar. Sadece "Timer1" SLEEP modunda iken harici cp ile çalıştırılabilir.

Örnek :

```
Uyu   SLEEP      ;PIC bu durumda çok az güç harcar. Arada bir kontrol
                 ;gereken güvenlik işlerinde, ya da belirli sürelerde
                 ;yapılacak işler bittiğinde PIC, uyuma moduna sokulur.
```

31-) SUBLW Bir sayı/sabitten W nin içeriğini çıkar

Söz dizim kuralı : [etiket] SUBLW k

İşleçler : $0 \leq k \leq 255$

işlevi : (k – W) → W

Status etkisi : C, DC, Z

Tanımı : k dan akümülatör içeriği çıkarılır.(İkiye tamamlama yöntemiyle). Sonuç W 'ye yüklenir.

Örnek :

```
SUBLW   h'02'      ;Komuttan önce W=h'01' ise, komuttan sonra W=01 h ve C=0
                   ;olur (sonuç pozitif).  W = 02 – 01 = 01.
SUBLW   h'01'      ; İkinci komut çalıştığında W=h'00' ve C=0 ve Z=1 olur.
                   ; (sonuç pozitif).  W = 01 – 01 = 00.
SUBLW   h'01'      ;W=h'02' olsun, 3.komutda çalıştığında W=h'FF' ve C=1 olur
                   ;(sonuç negatif).  W = 01 – 02 = FF ve C = 1.
```

32-) SUBWF f 'den W 'yı çıkar

Söz dizim kuralı : [etiket] SUBWF f,d

İşleçler : $0 \leq f \leq 127$ ve $d \in (1,0)$

işlevi : (f) - (W) → (Hedef)

Status etkisi : C, DC, Z

Tanımı : f yazmacının içeriğinden, W çıkarılır (İkiye tamamlama yöntemiyle). d=0 ise sonuç W ye, d=1 ise f yazmacına yüklenir.

Örnek :

```
SUBWF    f, 1      ; komuttan önce W=h'01' ve f=h'02' ise, komuttan sonra
                   ; f=01h ve C=0 olur (sonuç pozitif).
SUBWF    f, 0      ; ikinci komut çalıştığında W=h'00' ve C=0, Z=1 olur
                   ; (sonuç pozitif).
SUBWF    f, 1      ; Üçüncü komut da çalıştığında f=h'01've C=0 olur
                   ; (sonuç pozitif).
```

33-) SWAPF Yazmaç içeriğinde 4 'lülerin (digit) verini değiştir

Söz dizim kuralı : [etiket] SWAP f,d

işleçler : $0 \leq f \leq 127$ ve $d \in [0,1]$

işlevi : (f<3:0>) → (Hedef<7:4>) ve (f<7:4>) → (Hedef<3:0>)

Status etkisi : Yok

Tanımı : f yazmacının üst dörtlü biti ile alt dört biti yer değiştirirler. Sonuç d=0 ise W ye, d=1 ise f yazmacına yüklenir.

Örnek :

```
SWAPF   CAPRAZ,1    ; komutundan önce CAPRAZ=h'03', W=h'02' ise,
                    ; komuttan sonra CAPRAZ=h'30', W=h'02' olur.
SWAPF   CAPRAZ,0    ; komutundan önce CAPRAZ=h'03', W=h'02' ise,
                    ; komutu tekrarlanınca CAPRAZ=h'03', W=h'30' olur.
```

34-) XORLW Sayı ile W nin içeriğini XOR 'la

Söz dizim kuralı : [etiket] XORLW k

İşleçler : $0 \leq k \leq 255$

işlevi : (W) XOR k → (W)

Status etkisi : Z

Tanımı : W nin içeriği ile k literaline mantıksal XOR işlemi uygulanır. Sonuç W yazmacına yüklenir.

Örnek : XORLW h'03'

Komutundan önce W=h'01' ise; komuttan uygulandıktan sonra W=h'02' olur.

35-) XORWF Yazmaç içeriği ile W nin içeriğini XOR' la

Söz dizim kuralı : [etiket] XORWF f,d

işleçler : $0 \leq f \leq 127$ ve $d \in [0,1]$

işlevi : (W) XOR f → (Hedef)

Status etkisi : Z

Tanımı : W nin içeriği ile f yazmacına mantıksal XOR işlemi uygulanır. d=0 ise sonuç W yazmacına, d=1 ise f yazmacına yüklenir.

Örnek : XORWF f, 0

Komutundan önce f=h'0F', W=h'09' ise, komuttan sonra f=h'0F' ve W=h'06' olur.

G) MPLAB Programı ve Genel Özellikleri

MPLAB microchip firması tarafından geliştirilmiş olan bir programdır. MPLAB, assembly kodlarını yazmak için metin editörü, MPASM derleyicisi ve yazdığınız programı simülasyon yaparak görselleştirebildiğimiz MPSIM simülatörü gibi picsoftware için gerekli olan her şeyi üzerinde bulundurur. Bizim için önemli olan PIC' e .hex dosyasını yüklemektir. Ama yazdığımız programın doğru çalışıp çalışmadığını, hatalarını görüp düzeltmemizi sağlar. Çünkü her PIC üzerinde Flash bellek yoktur. Yani bazı PIC'ler bir kez programlanabilirler. Bu yüzden yazdığınız programın hatasız olması gerekir.

MPLAB programını www.microchip.com adresinde ücretsiz olarak indirebilirsiniz.

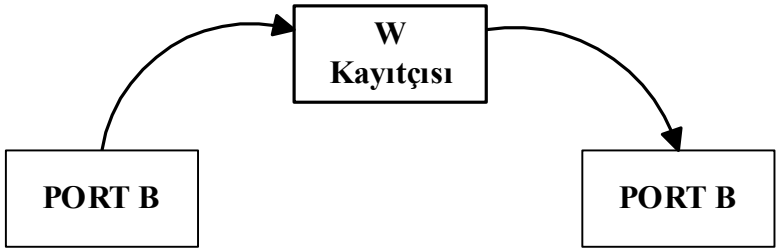
6.3. Bölüm Kaynakları

1. *O. Altınbaşak, 2001. "Mikrodenetleyiciler ve PIC Programlama", Atlaş Yayıncılık, İstanbul.*
2. *O. Urhan, M.Kemal Güllü, 2004. "Her Yönüyle PIC16F628", Birsen Yayınevi, İstanbul.*
3. *N. Topaloğlu, S. Görgünoğlu,2003. "Mikroişlemciler ve Mikrodenetleyiciler", Seçkin Yayıncılık, Ankara.*
4. *Y. Bodur, 2001. "Adım Adım PICmicro Programlama",İnfogate.*
5. *Ç. Akpolat, 2005."PIC Programlama", Pusula Yayıncılık.*

6.4. PIC PROGRAMLAMA

A) Veri Transferi

PIC içerisinde veri transferi işlemini kayıtçılar yardımıyla yaparız. W kayıtçısı, RAM bellek içerisindeki dosya kayıtçılarından bağımsız olarak bulunmakta ve veri transfer işlemi yapmada kullanılır. Örneğin; PORT B içerisinde var olan veriyi PORT A içerisine transfer etmek için aşağıdaki komutları yazmak gerekir.




```
MOVF    PORTB,W     ; PortB'nin içeriğini W kayıtçısına taşı
MOVWF   PORTA       ; W kayıtçısının içeriğini PortA'ya gönder
```

Örneğin; PortB'ye bağlı 8 adet LED bulunsun. Bu ledlerden ilk dört tanesini yakmak istersek aşağıdaki komutlar yazılmalıdır.

```
MOVLW   H'0F'       ; W kayıtçısına h'0F' yükle
MOVWF   PORTB       ;W kayıtçısının içeriğini PortB'ye gönderir.
```

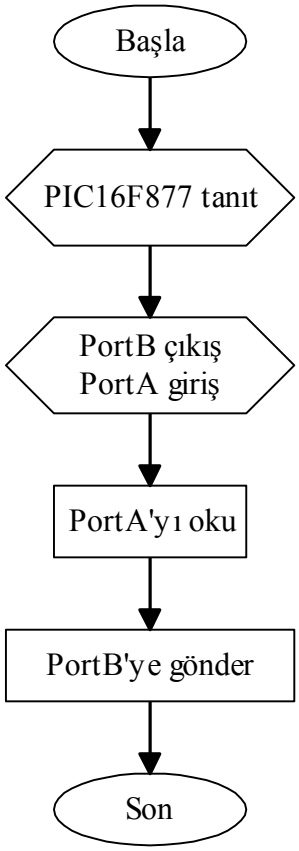
Burada W kayıtçısına gönderilen h '0F' verisini binary karşılığı b'00001111' dir. Bu veriye göre PortB'nin ilk 4 biti bağlı olan ledler yanar.

Eğer bir kayıtçının içerisine h '00' bilgisi gönderilmek isteniyorsa onun yerine CLRF komutu kullanılır. W kayıtçısının içeriği silinmek isteniyorsa da CLRW komutu kullanılır.

```
CLRF    PORTB       ; PortB nin içeriği temizlenir.
CLRW                ; W kayıtçısının içeriği temizlenir.
```

Örnek :

PortA'nın uçlarına bağlı olan butonlardan hangisi basılı tutulursa PortB'deki o butona karşılık gelen LED'i söndüren program ve akış şeması.

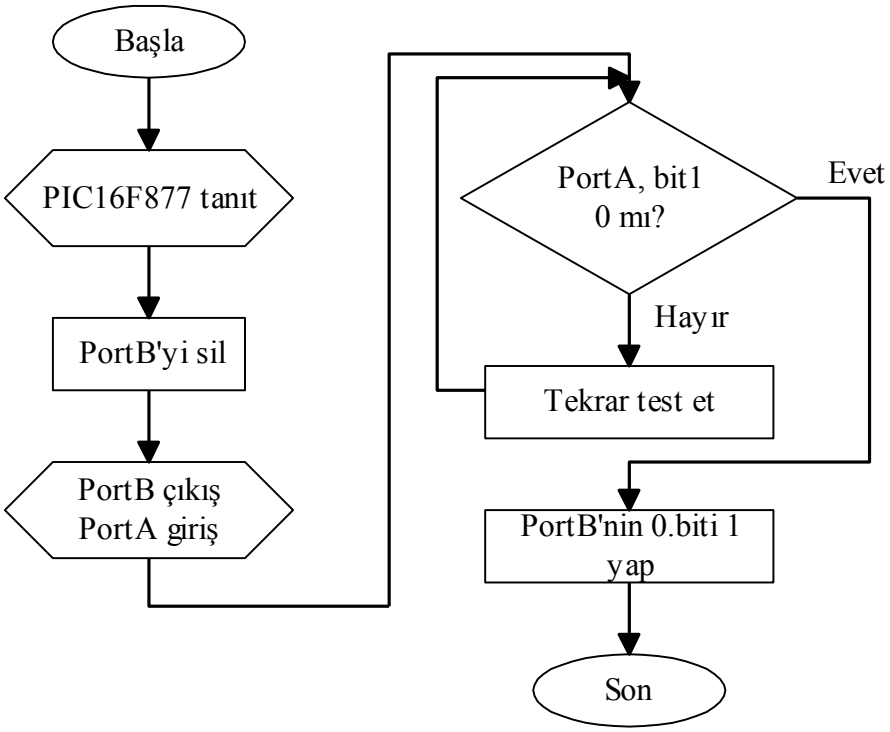



```
            LIST    P=16F877
PORTA       EQU     h ‘05’
PORTB       EQU     h ‘06’
STATUS      EQU     h ‘03’
TRISA       EQU     h ‘85’
TRISB       EQU     h ‘86’
            CLRF    PORTB       ; PortB’ye bağlı ledleri söndür
            BSF     STATUS,5    ; BANK1’e geç
            CLRF    TRISB       ; PortB’nin uçlarını çıkış yap
            MOVLW   h‘FF’       ; W kayıtçısına h‘FF’ i yükle
            MOVWF   TRISA       ; PortA’nın uçlarını giriş yap
            BCF     STATUS,5    ; BANK0’a geç
BASLA
            MOVF    PORTA,W     ; Porta’yı oku, sonucu W’ya yaz
            MOVWF   PORTB       ; Butonların durumunu PortB’de göster
DONGU
            GOTO    DONGU       ; Sonsuz döngü
            END                 ; Programın sonu
```

Duraklama komutu olmadığı için programda sonsuz döngü kullanılmıştır. Sonsuz döngü içerisine istenirse komut yazılabilir. Bu durumda reset tuşuna basılana kadar yada PIC’ın enerjisi kesilene kadar aynı komutlar tekrarlanır.

Bir kayıtçı içerisindeki herhangi bir biti test edilmek isteniyorsa BTFSC veya BTFSS komutları kullanılır. Bu test sonucuna göre program akışı istenilen komuta aktarılarak devam edilebilir.

Örnek : PortB'nin 0. bitine bağlı Led'i, A portunun 1. bitindeki butona basılınca yakan program.



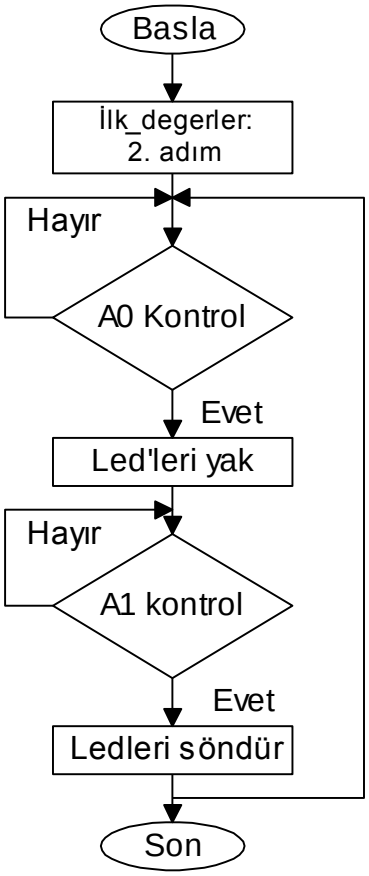

```
            LIST        P=16F877
PORTA       EQU         h '05'
PORTB       EQU         h '06'
STATUS      EQU         h '03'
TRISA       EQU         h '85'
TRISB       EQU         h '86'
            CLRF        PORTB       ; PortB'ye bağlı ledleri söndür
            BSF         STATUS,5    ; BANK1'e geç
            CLRF        TRISB       ; PORTB'nin uçlarını çıkış yap
            MOVLW       h 'FF'      ; W kayıtçısına h'FF' yükle
            MOVWF       TRISA       ; PortA'nın uçlarını giriş yap
            BCF         STATUS,5    ; BANK0'a geç
TEST_PORTA
            BTFSC       PORTA,1     ; A portunun 1. bitini test et
            GOTO        TEST_PORTA; 0 değilse tekrar test et
            BSF         PORTB,0     ; B portunun 0. bitini 1 yap
DONGU
            GOTO        DONGU
            END
```

Örnek : A portunun 2. bitindeki butona basınca B portuna bağlı tüm ledleri yakan program.

```
            LIST        P=16F877
PORTA       EQU         h '05'
PORTB       EQU         h '06'
STATUS      EQU         h '03'
TRISA       EQU         h '85'
TRISB       EQU         h '86'
            CLRF        PORTB       ; PortB'ye bağlı ledleri söndür
            BSF         STATUS,5    ; BANK1'e geç
            CLRF        TRISB       ; PORTB'nin uçlarını çıkış yap
            MOVLW       h 'FF'      ; W kayıtçısına h'FF' yükle
```

```
            MOVWF   TRISA       ; PortA'nın uçlarını giriş yap
            BCF     STATUS,5    ; BANK0'a geç
TEST_PORTA
            BTFSC       PORTA,2     ; A portunun 2. bitini test et
            GOTO    TEST_PORTA; 0 değilse tekrar test et
            MOVLW   h 'FF'      ; W kayıtçısına b '11111111' yükle
            MOVWF   PORTB       ; W içeriğini PortB'ye gönder.
DONGU
            GOTO    DONGU
            END
```

Örnek: PORTA 'nın 0. bitine bağlı butona basıldığında B portuna bağlı ledlerin yanmasını, 1. bitine bağlı butona basıldığında B portundaki ledleri söndüren program.

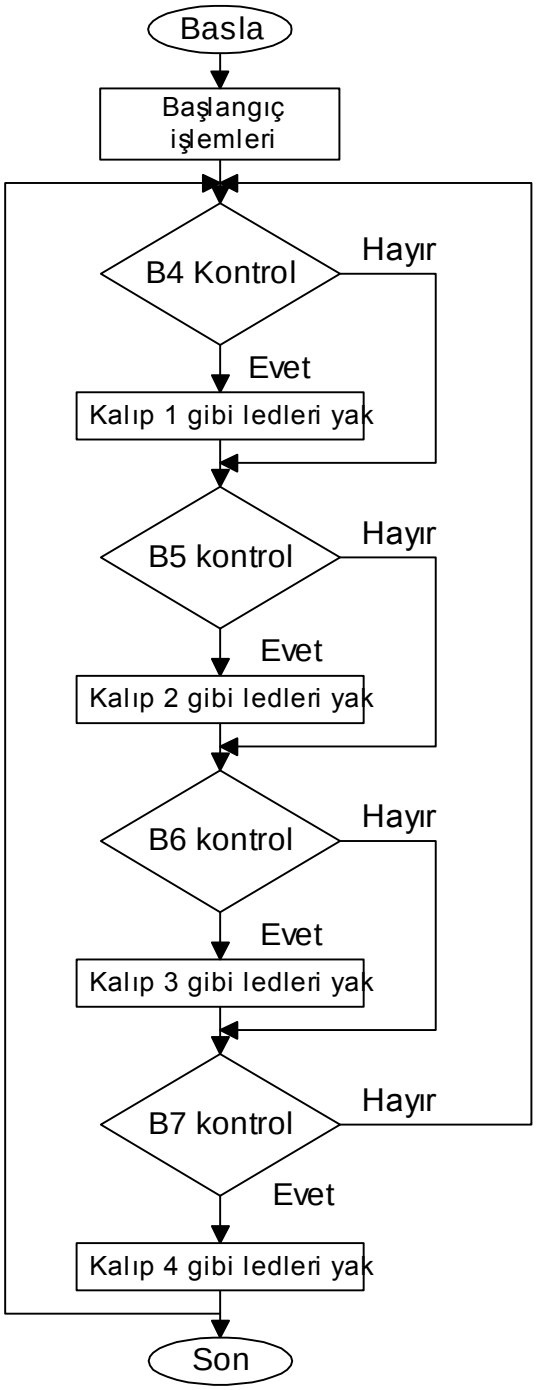



```
            LIST    p=16F877
            INCLUDE "p16F877"
            Org     0x00
            Goto    Basla
Basla
            CLRF    PORTA           ; PORTA yazmacını temizle
            CLRF    PORTB           ; PORTB yazmacını temizle
            BSF     STATUS,5        ; Bank1 e geç
            BCF     OPTION_REG,NOT_RBPU
            CLRF    TRISB           ; TRISB'yi temizle
            MOVLW   H '03'          ; A0 ve A1 giriş butonları bağlı
            MOVWF   TRISA
            BCF     STATUS,5
A0_KONTROL
```

```
            BTFSC       PORTA,0
            GOTO        A0_KONTROL
            MOVLW       H ‘FF’          ;Bütün Ledleri yak
            MOVWF       PORTB           ; tüm çıkış pinleri 1, ledleri yandı
A1_KONTROL
            BTFSC       PORTA,1
            GOTO        A1_KONTROL
            CLRF        PORTB
            GOTO        A0_KONTROL
            END
```

Örnek: PIC16F877 PortB ‘nin 4-7. bitlerine bağlı butonları kullanarak; PortD’ye bağlı sekiz ledi aşağıdaki yakan programı yazın.

B4 basılırsa: ○ ○ ● ● ● ● ● ●
B5 basılırsa: ● ● ○ ○ ● ● ● ●
B6 basılırsa: ● ● ● ● ○ ○ ● ●
B7 basılırsa: ● ● ● ● ● ● ○ ○
Yanan led ○ Sönen led ●



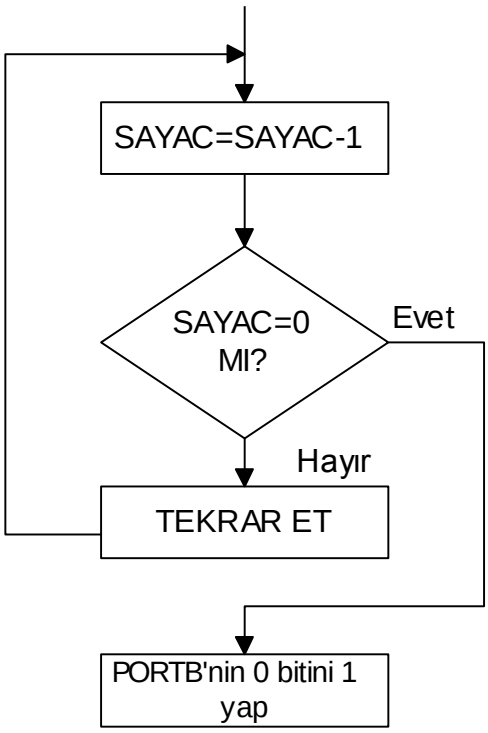
```
            LIST        p=16F877
            INCLUDE     “p16F877”
KALIP1      EQU         H ‘C0’          ; b ‘11000000’
KALIP2      EQU         H ‘30’          ; b ‘00110000’
KALIP3      EQU         H ‘0C’          ; b ‘00001100’
KALIP4      EQU         H ‘03’          ; b ‘00000011’
#DEFINE     B4          PORTB,4         ; Buton Adları
#DEFINE     B5          PORTB,5
#DEFINE     B6          PORTB,6
#DEFINE     B7          PORTB,7
            ORG         0X03
            GOTO        BASLA
BASLA
            CLRF        PORTB
            CLRF        PORTD
            BSF         STATUS,RP0
            BCF         OPTION_REG,7
            MOVLW       ‘F0’
            MOVWF       TRISB           ; PortB’nin 4-7 bitleri yönlendirildi.
            CLRF        TRISD           ; Portd çıkış
            BCF         STATUS,RP0
ANA_PROGRAM

B4_KONTROL
            BTFSC       B4              ; B4 butonuna basıldımı?
            GOTO        B5_Kontrol      ; hayır B5 butonunu kontrole git
            MOVLW       KALIP1
            MOVWF       PORTD
B5_KONTROL
            BTFSC       B5              ; B5 butonuna basıldımı?
            GOTO        B6_Kontrol      ; hayır B6 butonunu kontrole git
            MOVLW       KALIP2
            MOVWF       PORTD
B6_KONTROL
            BTFSC       B6              ; B6 butonuna basıldımı?
            GOTO        B7_Kontrol      ; hayır B7 butonunu kontrole git
            MOVLW       KALIP3
            MOVWF       PORTD
B7_KONTROL
            BTFSC       B7              ; B7 butonuna basıldımı?
            GOTO        B4_Kontrol      ; hayır B4 butonunu kontrole git
            MOVLW       KALIP4
            MOVWF       PORTD
            GOTO        ANA_PROGRAM
            END
```

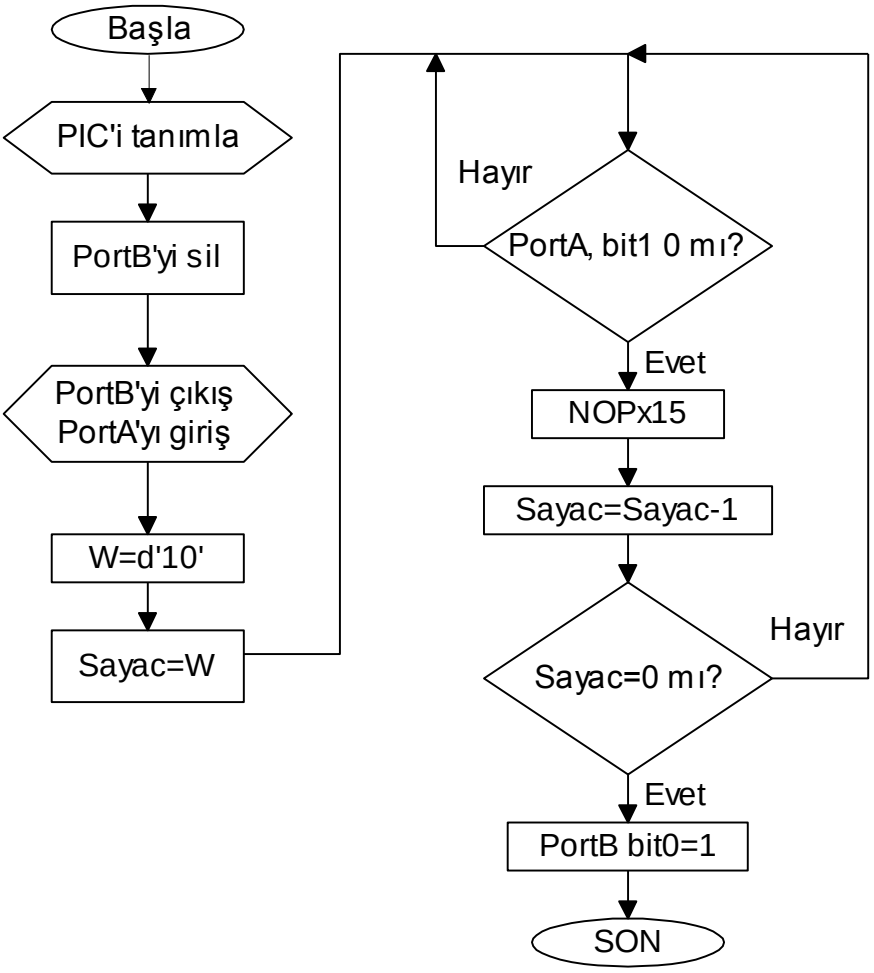
B) Döngü Düzenlemek

Program yazarken bazı işlemlerin belirli sayıda tekrarlanması gerekebilir. Bu durumda kayıtçılardan biri sayaç olarak kullanılır. Daha sonra her işlem tekrarlandığında sayaç değeri bir azaltılır. Azaltma işlemini DECFSZ komutu ile yapılır.

```
TEKRAR
        DECFSZ    SAYAC,F
        GOTO      TEKRAR
        BSF       PORTB,0
```

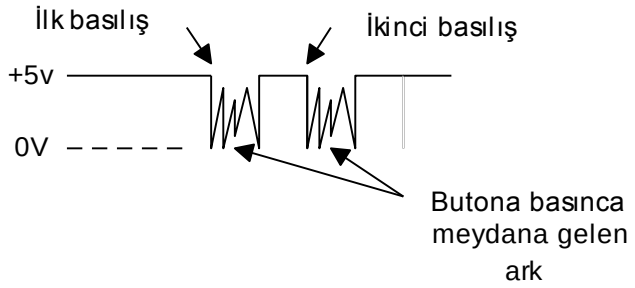


Örnek : A portunun 1. bitine bağlı butona 10 defa basıldıktan sonra B portunun 0. bitine bağlı olan ledleri yakan program ve akış şeması



```
        LIST    p=16F877
        INCLUDE "p16F877"
SAYAC   EQU     h '20'      ; Sayac isimli değişken tanımlanması
        CLRF    PORTB       ; PortB'ye bağlı ledleri söndür
        BSF     STATUS,5    ; Bank1 e geç
        CLRF    TRISB       ; PortB'nin uçlarını çıkış yap.
        MOVLW   h 'FF'      ; W kayıtçısına h 'FF' yüklenir
        MOVWF   TRISA       ; PortA'nın uçlarını giriş yap
        BCF     STATUS,5    ; Bank0 'a geç
BASLA
        MOVLW   d'10'       ; W kayıtçısına d'10' yükle
        MOVWF   SAYAC       ; SAYAC değişkenine W taşı
TEST
        BTFSC   PORTA,1     ; PortA'nın 1 biti 0 mı?
        GOTO    TEST        ; Değilse, TEST isimli etikete geri dön
        NOP
        NOP                 ; Gecikme zamanı için
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        DECFSZ  SAYAC,F     ; SAYAC değişkenin içeriği, 0 mı?
        GOTO    TEST        ; Değilse
        BSF     PORTB,0     ;PORTB'nin 0. bitini 1 yap
        END
```

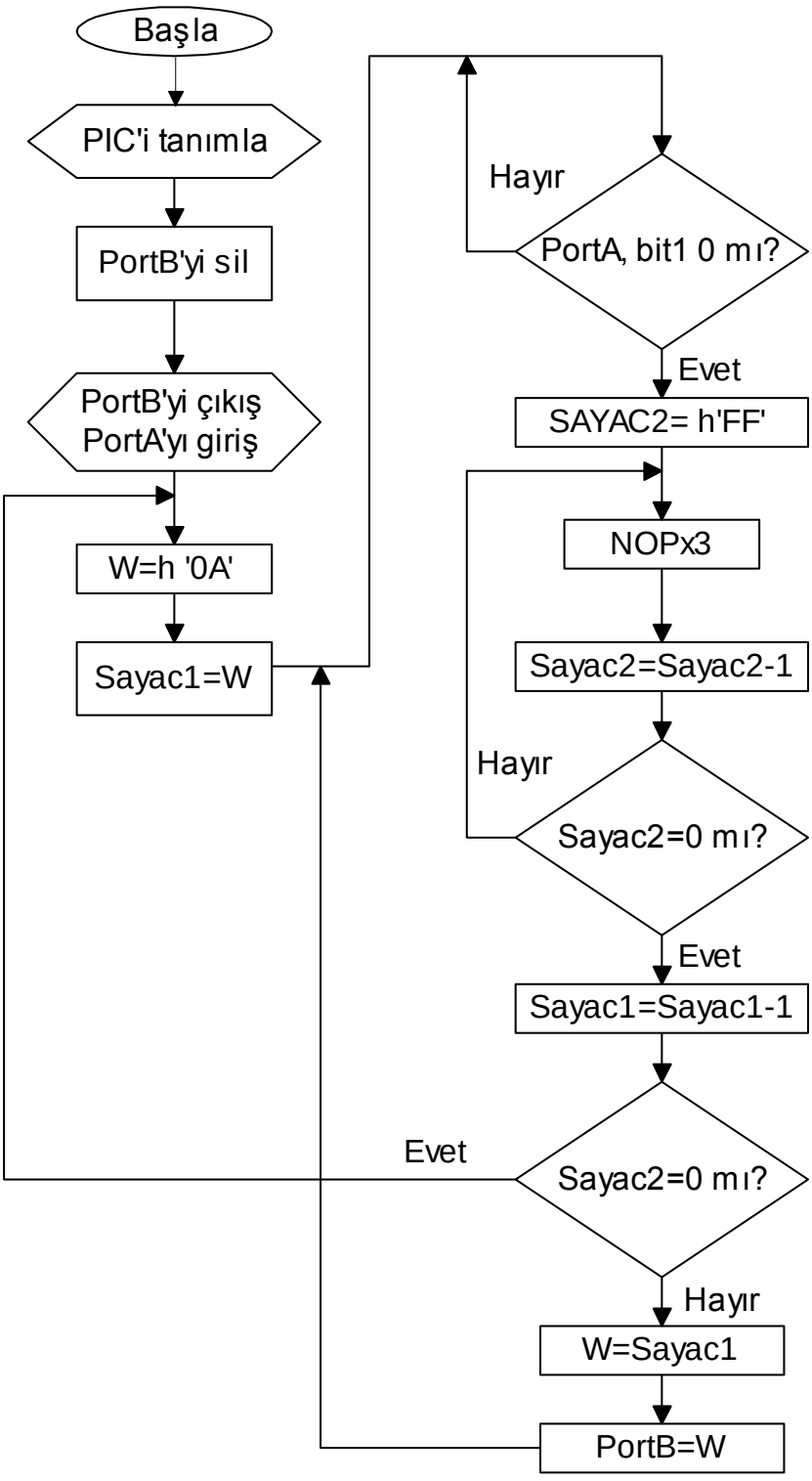
Butona basma sayısı 10'a ulaşmadan PortB'nin 0.bitindeki LED'in yandığı görülmektedir. Burada pull-up olayı gerçekleşmiştir. Yani butona basılmadığında +5 voltta basıldığında ise 0 V olmaktadır. Butona basma ve çekme esnasında bir ark oluşur.



Şekilde görüldüğü gibi butona basıldığı zaman gerilim dalgalanmaları yaşanacaktır. PIC komutlarının icra süreleri genelde 1 komut saykılında gerçekleşmektedir. 15 tane NOP komutu sadece 15 cplik bir zaman gecikmesi sağlamaktadır. Bu süre hesaplandığı zaman elimizi butondan çekmemizden daha kısa bir süreye denk gelmektedir. Her defasında 0 V seviyesine inişte butona

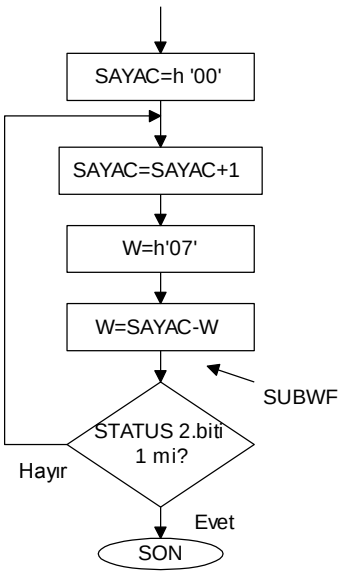
arka arkaya defalarca basılmış gibi işlem göstererek DECFSZ SAYAC,F komutuyla SAYAC kayıtçısının değeri her defasında 1 olacaktır. Butonun meydana getireceği ilk 0 seviyesine düşüşten sonraki 0 durumlarını eleyip tekrar 5V durumuna kadar belirli bir gecikme oluşturmak gerekecektir. Bu gecikmeyi NOP komutu kullanılarak oluşturulur. Programda kullandığımız 15 adet NOP komutu arkın etkilerini önlemekte yeterli olarak görülmediği takdirde sayısını fazlalaştırabiliriz.

Örnek: A portunun 1. bitindeki butona bastıkça B portundaki ledleri 9'dan 0'a kadar azaltarak yakan program ve akış şeması.



```
LIST        P16F877
            INCLUDE     "P16F877.INC"
SAYAC1      EQU         h '20'          ;SAYAC1 'e adres atandı
SAYAC2      EQU         h '21'          ;SAYAC2 'e adres atandı
            CLRF        PORTB           ; PortB ye bağlı ledleri söndür
            BSF         STATUS,5        ; Bank1'e geç
            CLRF        TRISB           ; PortB'nin uçlarını çıkış yap
            MOVLW       h 'FF'          ; W kayıtçısına h 'FF' yükle
            MOVWF       TRISA           ; Porta'nın uçlarını giriş yap
            BCF         STATUS,5        ;Bank0'a geç
BASLA
            MOVLW       h '0A'          ; W kayıtçısına h '0A' sayısını yükle
            MOVWF       SAYAC1          ; W içeriğini SAYAC1'e gönder
TEST
            BTFSC       PORTA,1         ; PortA'nın 1. biti 0 mı?
            GOTO        TEST            ; değilsei TEST isimli etikete geri dön
            MOVLW       h 'FF'          ; Evet ise, W'nın içeriğini h 'FF' yükle
            MOVWF       SAYAC2          ; W içeriğini SAYAC2 ye ata.
GECİKME
            NOP
            NOP                         ; gecikme işlemi
            NOP
            DECFSZ      SAYAC2,F        ; SAYAC2'nin içeriğini 0 olana kadar azalt
            GOTO        GECIKME
AZALT
            DECFSZ      SAYAC1,F        ;
            GOTO        YAK             ;SAYAC1=0 değilse,YAK isimli etikete geri dön
            GOTO        BASLA           ; SAYAC1=0 ise, BASLA isimli etikete geri dön
YAK
            MOVF        SAYAC1,W        ; SAYAC1'in içeriğini W'ya aktar.
            MOVWF       PORTB           ; W'ın içeriğini PortB'ye aktar
            GOTO        TEST            ; TEST isimli etikete geri dön
            END
```

Program yazarken bazı işlerin belirli sayılarda tekrarlanması istenebilir. Bu durumda da bir kayıtçı sayaç olarak kullanılır ve sayacın değeri her defasında 1 arttırılır. Arttırma işlemi INCF komutu ile yapılır. Sayaç belirlene değer ulaştığı zaman program akışı başka komuta geçer.




```
            CLRF        SAYAC
TEKRAR
            INCF        SAYAC,F
            MOVLW       h '07'
            SUBWF       SAYAC,W
            BTFSS       STATUS,2
            GOTO        TEKRAR
DONGU
            GOTO        DONGU
            END
```

C) Zaman Gecikmesi ve Alt Programlar

Bazı işlemlerin yapılması sırasında belirli bir zaman hiçbir şey yapmadan beklenmesi gerekir. Zaman geciktirme işlemlerini yazılım döngülerini kullanarak yapabildiğimiz gibi, donanımın bize sunduğu özel geciktirmeler yapabiliriz. Biz zaman geciktirme döngüsünde, gecikme zamanını tespit etmek için komutların çevrim süreleri dikkate alınır. RC osilatör kullanılan PIC devrelerinde bir komutun çevrim süresini hassas olarak hesaplamak kolay değildir. Ancak kristal veya seramik osilatör kullanılan devrelerde hassas gecikme döngüleri yapabiliriz.

PIC'in geciktirilmesi için ilk başta kullanıcılar NOP komutlarını kullanmayı tercih edebilirler.

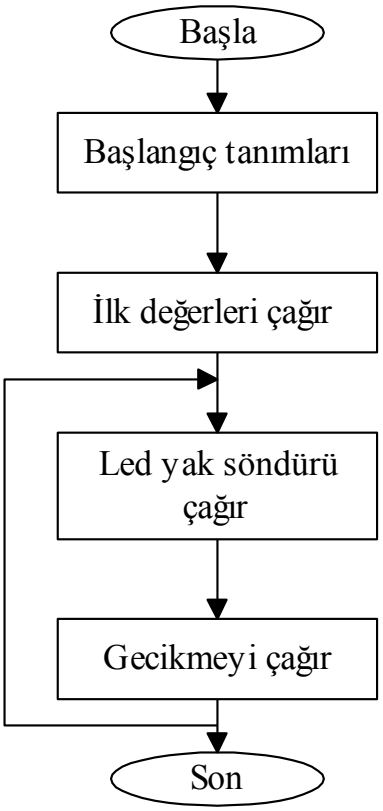
Örneğin NOP komutu ile 0,1 milisaniyelik bir gecikme yaratmak için ne kadar NOP komutu gerekir. (Kristal Osilatör → 20 Mhz)

PIC16F877 için Bir komutun çevrim süresi = $4 \times 0.05\mu sn = 0.2\ \mu sn$

Aynı gecikme için NOP komut sayısı ise $(0.1 \times 10^3)/(0.2) = 500$ adettir.

Bu sayıda NOP komutunun ardarda yazılması belleğin gereksiz biçimde dolmasına yol açar. Bu yöntem iyi bir programlama tekniği olarak da önerilmez. Bunun yerine daha az sayıda komut kullanarak, istenilen gecikmeyi sağlayabiliriz.

Örnek: PIC ile yapılan devrede çalışan ledleri 39 milisaniye aralılarla yakıp söndüren program ve akış şeması. (PORTD deki $RD_3 \ldots RD_0$ bağlı olan ledleri)



```
LIST        P16F877
            INCLUDE     "P16F877.INC"
SAYAC1      EQU         0x20
SAYAC2      EQU         0x21
            ORG         0x003
            GOTO BASLA
            ORG         0x004
DUR
            GOTO DUR
ILK_DEGERLER
            CLRF        PORTD       ; PortD yazmanını temizle
            BCF         STATUS,6    ; Bank1'e geç
            BSF         STATUS,5    ; Bank1'e geç
            CLRF        TRISD       ; D Portundaki ledler çıkış seçilir
            BCF         STATUS,5    ; Bank0'a geç
            RETURN
TEKRAR_YAK
            CLRF        PORTD       ; Ledleri söndür
            CALL        GECIKME     ; Gecikme alt programını çağırır
            MOVLW       b'00001111' ; W kayıtçısına b'00001111' değeri yüklendi
            MOVWF       PORTD       ; W içeriği PortD ye yüklendi
            RETURN
GECİKME
            MOVLW       h 'FF'
            MOVWF       SAYAC1      ; SAYAC1= d'255'
DONGU11
            MOVLW       h'FF'
            MOVWF       SAYAC2      ; SAYAC2= d'255'
DONGU12
            DECFSZ      SAYAC2,F
            GOTO        DONGU12
            DECFSZ      SAYAC1,F
            GOTO DONGU11
            RETURN
BASLA
            CALL        ILK_DEGERLER
DONGU
            CALL        TEKRAR_YAK
            CALL        GECİKME
            GOTO        DONGU
            END
```

Gecikme programında iç içe iki döngü kullanılmıştır. Her iki döngü kullanıldığında oluşan toplam komut çevrim sürelerini hesaplarsak;

KOMUTLAR			KOMUT ÇEVRİM SÜRESİ
GECİKME			
	MOVLW h 'FF'		1
	MOVWF SAYAC1	; d'255'=M	1
DONGU11			
	MOVLW h'FF'		1xM
	MOVWF SAYAC2	; d'255'=N	1xM
DONGU12			
	DECFSZ SAYAC2,F		1xMxN
	GOTO DONGU12		2xMxN
	DECFSZ SAYAC1,F		1xM
	GOTO DONGU11		2xM
	RETURN		2

M ve N yerine 255 yerleştirilirse; Toplam 196.608 çevrim süresi

196.608 x (0.05 x 4)µsn = 39.321 µsn ≡39 msn

Gecikme sürelerini sayaçlara yüklediğimiz M ve N sabitlerini değiştirerek ayarlamak mümkündür. Örneğin gecikme süresinin 10 milisn olması için dış ve iç döngü sayaçlarının değerlerinin ne olması gerektiğini bulalım:

10.000/(0.05 x 4) = 3xNxN N=129

bu değer h'81' değerine karşılık gelir. Böylece 10 milisn gecikme elde edilir.

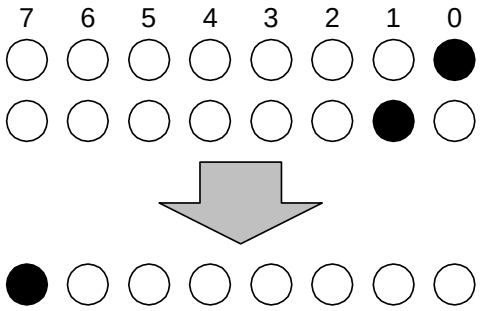
Örnek: PortA'nın 1. ucuna bağlı butona 10 defa basıldıktan sonra PortB deki tüm ledleri yakan program.

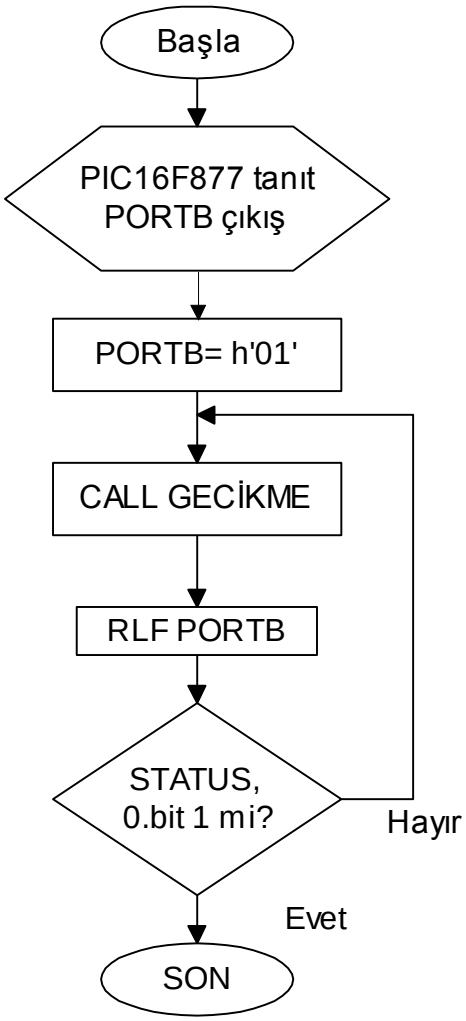
```
            LIST    P16F877
            INCLUDE "P16F877.INC"
SAYAC1      EQU     h '20'
SAYAC2      EQU     h '21'
MEM         EQU     h '22'
            CLRF    PORTB     ; PortB'yi sil
            BSF     STATUS,5  ; Bank1'e geç
            CLRF    TRISB     ; PortB'nin uçları çıkış
            BSF     TRISA,1   ; Porta'nın 1. biti giriş
            CLRF    MEM       ; MEM kayıtçısını temizle
TEKRAR
            BTFSC   PORTA,1   ; PortA'nın 1.bit'i 0 mı?
            GOTO    TEKRAR    ; Hayır tekrar test et
            INCF    MEM       ; Evet, MEM=MEM+1
            MOVF    MEM,W     ; W=MEM
            SUBLW   d '10'    ; W= d '10'-W
            BTFSC   STATUS,2  ; STATUS'un 2.biti 0 mı?
            GOTO    YAK       ; Hayır, Z=1
            CALL    GECİKME   ; Gecikme altprogramını çağır
            GOTO    TEKRAR    ;
YAK
            MOMVLW  h 'FF'    ; W= h 'FF'
            MOVWF   PORTB     ; PortB'deki tüm ledleri yak
DONGU
            GOTO    DONGU     ;
;**************GECİKME ALTPROGRAMI**********
GECIKME
            MOVLW   h 'FF'
            MOVWF   SAYAC1
DONGU1
            MOVLW   h 'FF'
            MOVWF   SAYAC2
DONGU2
            DECFSZ  SAYAC2,F
            GOTO    DONGU2
            DECFSZ  SAYAC1,F
            GOTO    DONGU1
            RETURN
            END
```

D) Bit Kaydırma

Bit kaydırma komutları RLF, RRF, COMF ve SWAPF komutlarıdır. Bı komutlar kullanılarak farklı uygulamalar yapılabilmektedir.

Örnek : PortB'ye bağlı 8 led üzerindeki bir ledin yanışını LED0'dan LED7'ye doğru kaydıran program ve akış şeması.



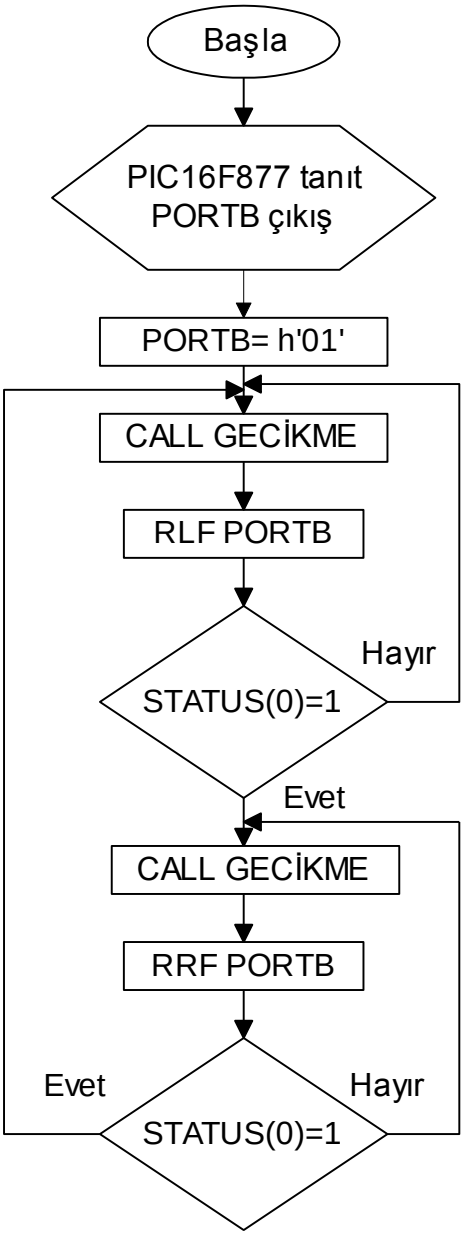



```
            LIST        P16F877
            INCLUDE     "P16F877.INC"
SAYAC1      EQU         h '20'          ;SAYAC1 'e adres atandı
SAYAC2      EQU         h '21'          ;SAYAC2 'e adres atandı
            CLRF        PORTB           ; PortB ye bağlı ledleri söndür
            BCF         STATUS,0        ; Carry flag'ı sıfırla
            BSF         STATUS,5        ; Bank1'e geç
            CLRF        TRISB           ; PortB'nin uçlarını çıkış yap
            BCF         STATUS,5        ;Bank0'a geç

            MOVLW       h '01'          ; b '00000001' sayısını W'ya yükle
            MOVWF       PORTB           ; W kayıtçısının içeriğini PortB'ye yükle
TEKRAR
            CALL        GECIKME         ; Gecikme yap
            RLF         PORTB,F         ; PortB'deki veriyi sola kaydır
            BTFSS       STATUS,0        ; Carry flag 1 mi?
```

```
            GOTO        TEKRAR      ; Hayır
DONGU
            GOTO        DONGU       ;
GECİKME
            MOVLW       h ‘FF’
            MOVWF       SAYAC1      ; SAYAC1= d’255’
DONGU11
            MOVLW       h’FF’
            MOVWF       SAYAC2      ; SAYAC2= d’255’
DONGU12
            DECFSZ      SAYAC2,F
            GOTO        DONGU12
            DECFSZ      SAYAC1,F
            GOTO        DONGU11
            RETURN
            END
```

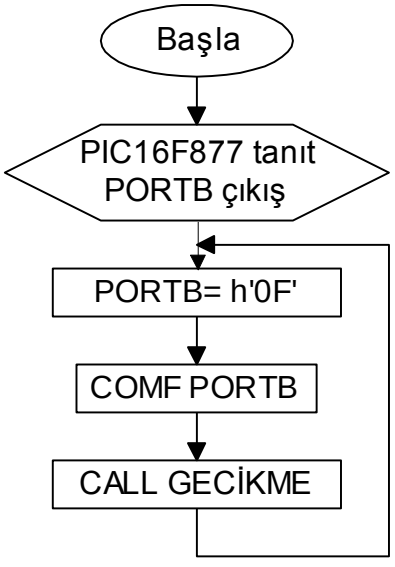
Örnek : PortB’ye bağlı olan 8 LED üzerinde bir LED’in yanışını sağa-sola kaydıran ve bu işlemi sürekli tekrarlayan program ve akış şeması.**(Karaşimşek devresi)**



```
            LIST        P16F877
            INCLUDE     "P16F877.INC"
SAYAC1      EQU         h '20'          ;SAYAC1 'e adres atandı
SAYAC2      EQU         h '21'          ;SAYAC2 'e adres atandı
            CLRF        PORTB           ; PortB ye bağlı ledleri söndür
            BCF         STATUS,0        ; Carry flag'ı sıfırla
            BSF         STATUS,5        ; Bank1'e geç
            CLRF        TRISB           ; PortB'nin uçlarını çıkış yap
            BCF         STATUS,5        ;Bank0'a geç

            MOVLW       h '01'          ; b '00000001' sayısını W'ya yükle
            MOVWF       PORTB           ; W kayıtçısının içeriğini PortB'ye yükle
SOL
            CALL        GECIKME         ; Gecikme yap
            RLF         PORTB,F         ; PortB'deki veriyi sola kaydır
            BTFSS       STATUS,0        ; Carry flag 1 mi?
            GOTO        SOL             ;
SAG
            CALL        GECIKME         ; Gecikme yap
            RRF         PORTB,F         ; PortB'deki veriyi sağa kaydır
            BTFSS       STATUS,0        ; Carry flag 1 mi?
            GOTO        SAG             ;
            GOTO        SOL             ;
GECİKME
            MOVLW       h 'FF'
            MOVWF       SAYAC1          ; SAYAC1= d'255'
DONGU1
            MOVLW       h'FF'
            MOVWF       SAYAC2          ; SAYAC2= d'255'
DONGU2
            DECFSZ      SAYAC2,F
            GOTO        DONGU2
            DECFSZ      SAYAC1,F
            GOTO  DONGU1
            RETURN
            END
```

Örnek : PortB'deki LED'leri dönüşümlü olarak ilk önce ilk dört biti, daha sonrada son dört bitteki ledleri yakan program ve akış şeması.



```
            LIST        P16F877
            INCLUDE     "P16F877.INC"
SAYAC1      EQU         h '20'          ;SAYAC1 'e adres atandı
SAYAC2      EQU         h '21'          ;SAYAC2 'e adres atandı
            CLRF        PORTB           ; PortB ye bağlı ledleri söndür
            BSF         STATUS,5        ; Bank1'e geç
            CLRF        TRISB           ; PortB'nin uçlarını çıkış yap
            BCF         STATUS,5        ;Bank0'a geç
            MOVLW       h '0F'          ; b '00001111' sayısını W'ya yükle
            MOVWF       PORTB           ; W kayıtçısının içeriğini PortB'ye yükle
TERSLE
            COMF        PORTB,F         ; PortB'deki veriyi tersle
            CALL        GECIKME         ; Gecikme yap
            GOTO        TERSLE
GECİKME
            MOVLW       h 'FF'
            MOVWF       SAYAC1          ; SAYAC1= d'255'
DONGU1
            MOVLW       h'FF'
            MOVWF       SAYAC2          ; SAYAC2= d'255'
DONGU2
            DECFSZ      SAYAC2,F
            GOTO        DONGU2
            DECFSZ      SAYAC1,F
            GOTO        DONGU1
            RETURN
            END
```

E) Çevrim Tabloları

Çevrim tabloları bir kodu başka bir koda çevirmek için kullanılırlar. Örneğin PORTB'ye bağladığımız 7 segment display'in üzerindeki heksadesimal karakterleri görmek istiyoruz. Çevrim tablosuna yerleştirdiğimiz heksadesimal koda karşılık gelen uygun kodu seçip, çıkışa göndermemiz gerekir.

Çevrilecek kod. Hex. sayı	Çevrilen 7 segment kodu (PORTB'wye)	7 segment uçlarındaki veri	7 segment 'te görülecek sayı
h '00'	h '3F'	00111111	0
h '01'	h '06'	00000110	1
h '02'	h '5B'	01011011	2
h '03'	h '4F'	01001111	3
h '04'	h '66'	01100110	4
h '05'	h '6D'	01101101	5
h '06'	h '7D'	01111101	6
h '07'	h '07'	00000111	7
h '08'	h '7F'	01111111	8
h '09'	h '6F'	01101111	9
h '0A'	h '77'	01110111	A
h '0B'	h '7C'	01111100	B
h '0C'	h '39'	00111001	C
h '0D'	h '5E'	01011110	D
h '0E'	h '79'	01111001	E
h '0F'	h '71'	01110001	F
Nokta	h '80'	10000000	.

Örnek : 7 segmentli display üzerinde “5” sayısını gösteren program.

```
            LIST    P16F877
            INCLUDE “P16F877.INC”
            CLRF    PORTB       ; PortB ye bağlı ledleri söndür
            BSF     STATUS,5    ; Bank1’e geç
            CLRF    TRISB       ; PortB’nin uçlarını çıkış yap
            BCF     STATUS,5    ;Bank0’a geç
BASLA
            MOVLW   h ‘05’
            CALL    TABLO
            MOVWF   PORTB
DONGU
            GOTO    DONGU
TABLO
            ADDWF   PCL,F       ; PCL ← W( h ’05’)
            RETLW   h ‘3F’
            RETLW   h ‘06’
            RETLW   h ‘5B’
            RETLW   h ‘4F’
            RETLW   h ‘66’
            RETLW   h ‘6D’      ; W ← h ‘6D’
            RETLW   h ‘7D’
            RETLW   h ‘07’
            RETLW   h ‘7F’
            RETLW   h ‘6F’
            RETLW   h ‘77’
            RETLW   h ‘7C’
            RETLW   h ‘39’
            RETLW   h ‘5E’
            RETLW   h ‘79’
            RETLW   h ‘71’
            RETLW   h ‘80’
            END
```

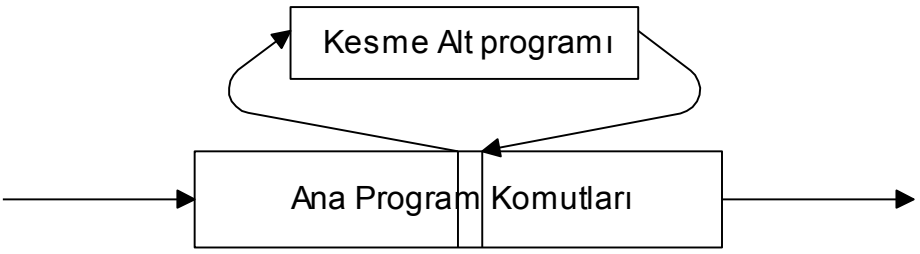
Örnek : PORTB’nin uçlarına bağlı 7 segment display’de 0~F arasında saydıran program.

```
            LIST    P16F877
            INCLUDE “P16F877.INC”
SAYAC1      EQU     h ‘20’      ;SAYAC1 ‘e adres atandı
SAYAC2      EQU     h ‘21’      ;SAYAC2 ‘e adres atandı
SAYAC       EQU     h ‘22’
            CLRF    PORTB       ; PortB ye bağlı ledleri söndür
            BSF     STATUS,5    ; Bank1’e geç
            CLRF    TRISB       ; PortB’nin uçlarını çıkış yap
            BCF     STATUS,5    ;Bank0’a geç
BASLA
            MOVLW   h ‘00’      ; b ‘00000000’ sayısını W’ya yükle
            MOVWF   SAYAC       ; W kayıtçısının içeriğini SAYAC’a yükle
DONGU
            MOVF    SAYAC,W     ; W ← SAYAC
            ANDLW   B ‘00001111’ ; W’nin üst dört bitini sıfırla
            CALL    CEV_TAB     ; çevrim tablosunu çağır
            MOVWF   PORTB       ; kodu 7 segmentte göster
            INCF    SAYAC,F     ; SAYAC ←SAYAC+1
```

```
            CALL        GECIKME     ;
            GOTO        DONGU
CEV_TAB
            ADDWF       PCL,F       ; PCL ← W( h '05')
            RETLW       h '3F'      ; 0
            RETLW       h '06'      ; 1
            RETLW       h '5B'      ; 2
            RETLW       h '4F'      ; 3
            RETLW       h '66'      ; 4
            RETLW       h '6D'      ; 5
            RETLW       h '7D'      ; 6
            RETLW       h '07'      ; 7
            RETLW       h '7F'      ; 8
            RETLW       h '6F'      ; 9
            RETLW       h '77'      ; A
            RETLW       h '7C'      ; B
            RETLW       h '39'      ; C
            RETLW       h '5E'      ; D
            RETLW       h '79'      ; E
            RETLW       h '71'      ; F
GECİKME
            MOVLW       h 'FF'
            MOVWF       SAYAC1
DONGU1
            MOVLW       h'FF'
            MOVWF       SAYAC2
DONGU2
            DECFSZ      SAYAC2,F
            GOTO        DONGU2
            DECFSZ      SAYAC1,F
            GOTO        DONGU1
            RETURN
            END
```

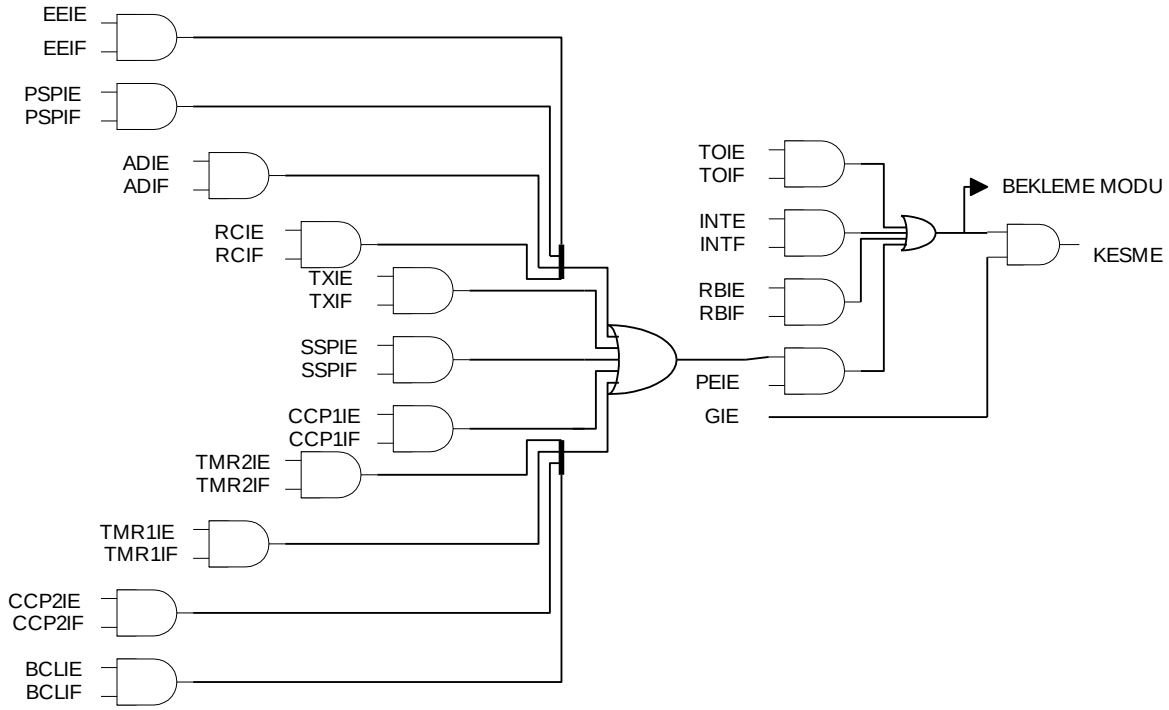
F) Kesmeler

PIC'in port girişlerinden veya donanım içerisindeki bir sayıcıdan gelen sinyal nedeniyle belleğinde çalışmakta olan programın kesilmesi olayı kesme (interrupts) olarak adlandırılır. Program kesildiği andan hemen sonra ana program kaldığı yerden itibaren tekrar çalışmasına devam eder. Kesme işlemi ana programın çalışmasını sadece duraklatır. Ana programın çalışma işlevini devam ettirmesini engellemez.



Şekil 9.1. Kesme programının çalışması

Kesme ile alt programın karıştırılmamsı gerekir. İlk bakışta arada fark yokmuş gibi olsa da farklıdır. Normal alt programı çağırma CALL komutu ile yapılır. Ancak kesme alt programlarının çağrılması ise donanımda oluşan değişiklikler sonucunda olur. Bir kesme meydana geldiğinde o anda çalışmakta olan komut çalışmasını tamamlar. Daha sonra program PIC program belleğinin h '0004' adresine atlar ve bu adresteki komutu çalıştırmaya başlar. PIC kesme alt programı çalıştıktan sonra ana program hangi adrese geri gideceğini yığına kaydeder. Kesme alt programından ana programa dönüş komutu olarak RETFIE komutu kullanılır.

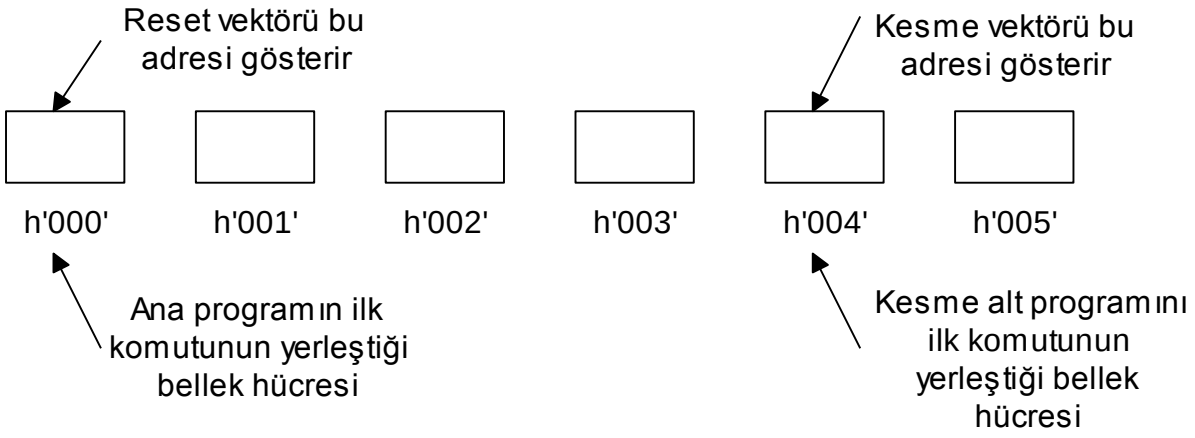


Device	TOIF	INTF	RBIF	PSPIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	EEIF	BCLIF	CCP2IF
PIC16F876-873	+	+	+	-	+	+	+	+	+	+	+	+	+	+
PIC16F877-874	+	+	+	+	+	+	+	+	+	+	+	+	+	+

Bir kesme olayı meydana geldiğinde;

- Kesme olayı meydana geldiğinde yığın(stack) kayıtçısının olduğu adrese (h '23F') atlanır.
- Ana programın kaldığı adres yığına kaydedilir.
- h '04' adresindeki komut çalıştırılır.
- Kesme alt programının olduğu adrese atlanır.
- Kesme alt programını çalıştırılır.
- Yığına geri dönülür.
- Ana programın kaldığı yerin adresi alınır.
- Ana programın çalışmasına devam edilir.

Kesme kullanılmadığı zaman ana program, program belleğinin h ‘0000’ adresinden itibaren h‘0004’ adresine doğru herhangi bir karışıklığa sebep olmadan çalışır. Kesme kullanılacaksa programcı tarafından başka bir çalışma sırası düzenlemesi gerekecektir.



Şekil . Kesme vektörün düzenlenmesi

Programın düzenlenmesi ise şu şekilde olmalıdır.

```
        ORG         h ‘000’
        GOTO        BASLA       ; Ana program başlangıcı
        ORG         h ‘004’
        GOTO        KESME_PROG      ;kesme alt program başlangıcı
BASLA
        Ana program komutları
        …
        …
        …
KESME_PROG
        Kesme alt program komutları
        …
        …
        RETFIE
```

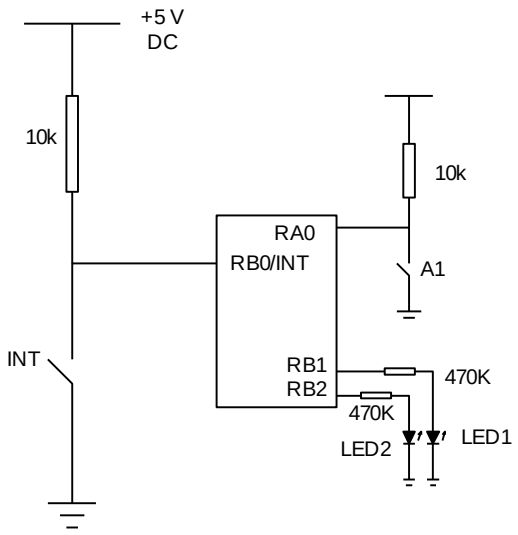
Bir kesme oluştuğu zaman kesme alt programı çalışmadan önce gecikme meydana gelir. Bu gecikme süresi 3 yada 4 komut saykılı süresindedir. Zamanlamanın çok önemli olduğu uygulamalarda bu zaman gecikmesi dikkate alınmalıdır.

Kesme oluştuğunda, kesme altprogramına sapılır. Kesmenin oluştuğu sırada önce işlenmekte olan komut tamamlanır. Komut adresi aynı altprogramlardaki gibi yığının tepesine yerleştirilir. Bu adresteki GOTO komutu ise kesme yordamına sapmayı sağlar. Kesme altprogramının komutları işlenir ve yığına konmuş olan adres, program sayacına aktarılarak, kesme oluştuğu sırada işlenen komutun adresi bulunur. Bundan sonra program sayacının değeri bir arttırılır ve program kaldığı yerden devam eder.

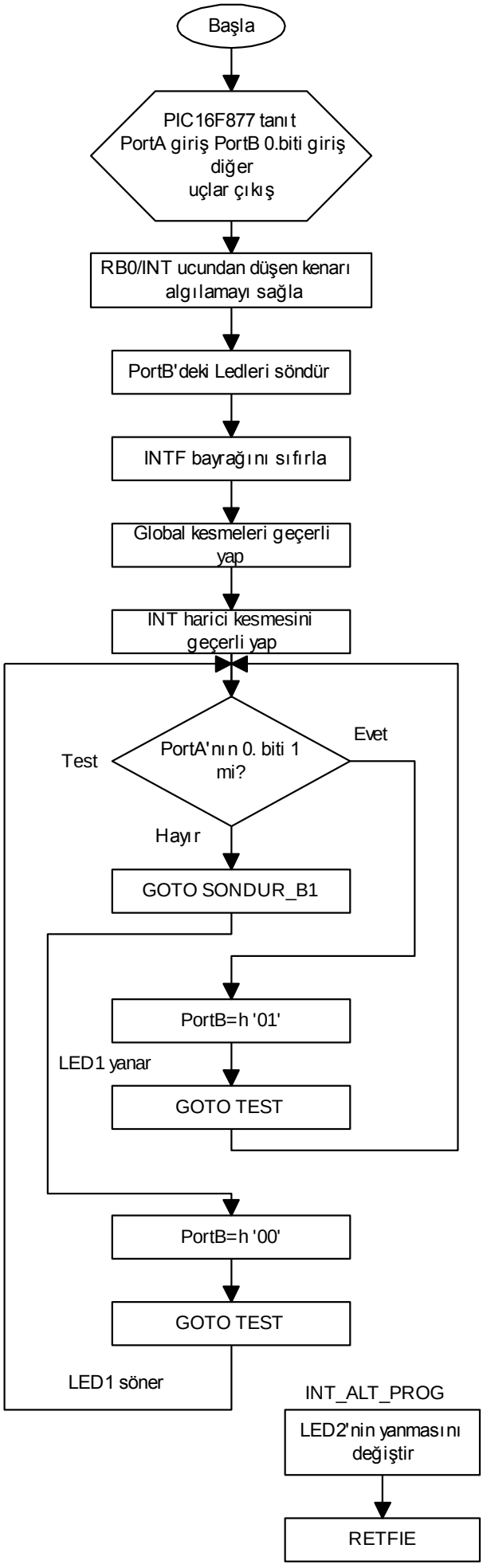
Kesme kullanılırken; PCL, Status ve W yazmaçlarının içindeki değerleri koruma işi de programlayıcıya bırakılmıştır. Bu yazmaçları korumak için PIC veri sayfalarında aşağıdaki komutları kullanmak yeterlidir.

```
MOVWF     W_TEMP          ; W geçici değişkene kopyala
SWAPF     STATUS,W        ; Status'u SWAP ile W'ye yükle
CLRF      STATUS          ; IRP, RP1 ve RP0'ı temizle
MOVWF     STATUS_TEMP     ; Status'u Bank0'da geçici değişkene yükle
MOVF      PCLATH,W        ;
MOVWF     PCLATH_TEMP     ; Geçici PCLATH'ı W yazmacına yükle
CLRF      PCLATH          ; PCLATH'ı temizle
…….
Komutlar
…….
MOVF      PCLATH_TEMP,W   ;
SWAPF     STATUS_TEMP,W   ; STATUS_TEMP'i W'ya yükle
MOVF      STATUS          ; W yazmacını Status'a aktar
SWAPF     W_TEMP,F        ; W_TEMP'e SWAP uygula
SWAPF     W_TEMP,W        ; W_TEMP'e 2. kez SWAP uygula
```

Örnek : RB0/ INT ucundan girilen bir sinyal ile kesme oluşturmaya çalışınız.



Şekilde görüldüğü gibi bu programın amacı PORTA nın 0.bitine bağlı olan butonun basılı olup olmadığı gösteren basit bir programdır. RB0/INT ucundan bir sinyal girerek kesme oluşturmak için RB0 ucuna bir buton bağlanmıştır. INT butonuna basılınca kesme oluşur ve kesme alt programı çalışarak LED2 in yanma durumunda değişiklik görülür.


Başla
PIC16F877 tanıt
PortA giriş PortB 0.biti giriş
diğer
uçlar çıkış
RB0/INT ucundan düşen kenarı
algılamayı sağla
PortB'deki Ledleri söndür
INTF bayrağını sıfırla
Global kesmeleri geçerli
yap
INT harici kesmesini
geçerli yap
Test
PortA'nın 0. biti 1
mi?
Evet
Hayır
GOTO SONDUR_B1
PortB=h '01'
LED1 yanar
GOTO TEST
PortB=h '00'
GOTO TEST
LED1 söner
INT_ALT_PROG
LED2'nin yanmasını
değiştir
RETFIE

```
            LIST        P16F877
            INCLUDE     "P16F877.INC"
            ORG         h '000'
            GOTO        BASLA
            ORG         h '004'
            GOTO        INT_ALT_PROG
BASLA
            BSF         STATUS,5      ; bank1 e geç
            MOVLW       h 'FF'        ; W←h 'FF'
            MOVWF       TRISA         ; PortA giriş
            MOVLW       b '00000001'  ; W←h '01'
            MOVWF       TRISB         ; PortB 0. bit giriş
            MOVLW       b '10111111'  ; W← b '10111111' düşen kenar
            MOVWF       OPTION_REG; W yı Option kayıtçısına yükle
            BCF         STATUS,5      ; Bank0 a geç
            CLRF        PORTB         ; PortB'yi sil
            BCF         INTCON,1      ; INTF bayrağını sil, kesmeyi hazırla
            BSF         INTCON,7      ; Global kesmeyi aktif yap
            BSF         INTCON,4      ; RB0/INT kesmesini geçerli yap
TEST
            BTFSC PORTA,0       ; PortA'nın 1. bitini test et
            GOTO  SONDUR_LED1
YAK_LED1
            BSF         PORTB,1       ; PortB'nin 1. bitini 1 yap
            GOTO        TEST
SONDUR_LED1
            BCF         PORTB,1       ; PortB'nin 1. bitini 0 yap
            GOTO  TEST
INT_ALT_PROG
            BCF         INTCON,1      ; INTF bayrağını sil
            MOVLW       b '00000100'  ; terslenecek olan biti W'ya  yükle
            XORWF       PORTB,F       ; RB2' yi tersle
            RETFIE                    ; Kesme alt programından dön.
            END
```

RB0 ucundan girilen sinyalin düşen kenarında kesmenin oluşması programda

```
            MOVLW       h '10111111'
            MOVWF       OPTION_REG
```

komutları kullanılmıştır. INTCON kayıtçısının 7 biti GIE bayrağının bulunduğu bittir. Burada tüm kesme işlemleri aktif duruma getirilmiştir. Bu kayıtçının 4. biti INTE bayrağının bulunduğu bittir ve harici kesmeyi aktif yapmayı sağlar. 1. bit ise harici kesme bayrağıdır. 0 ise kesme var 1 ise kesme yoktur. PIC'de yazılan programlar mikrodenetleyicinin işlem yapma gücü artmaktadır.

G) Zamanlayıcılar

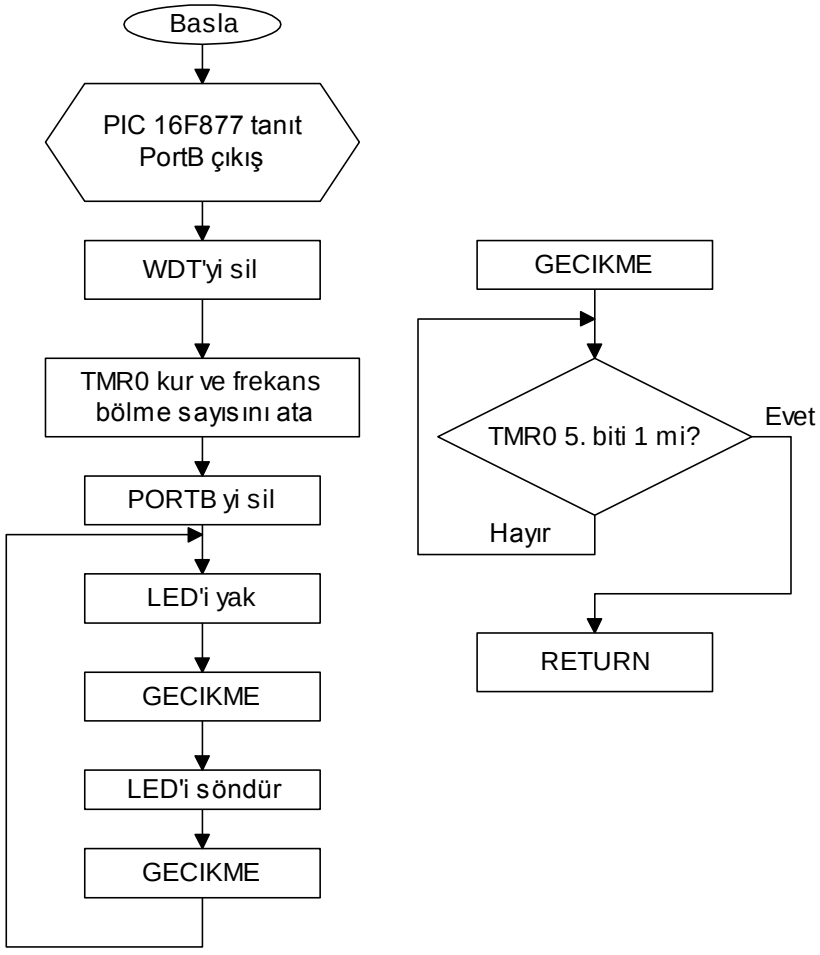
PIC16F877 ailesinde Timer0, Timer1, Timer2 ve WDT adları verilen 4 tip zamanlayıcı bulunmaktadır.

Timer0 dış olayların sayılmasında ve istenen sayıda dış olay meydana geldiğinde, kesme oluşturmakta kullanılır. İçinde PIC' in kendi hızından daha hızlı, 50Mhz 'e kadar varan hızlara uyum sağlayıp haberleşebilecek önbölücüler bulunur. Timer0 istenirse kendi iç kristal saatinide kullanabilir. Timer0 temel özellikleri;

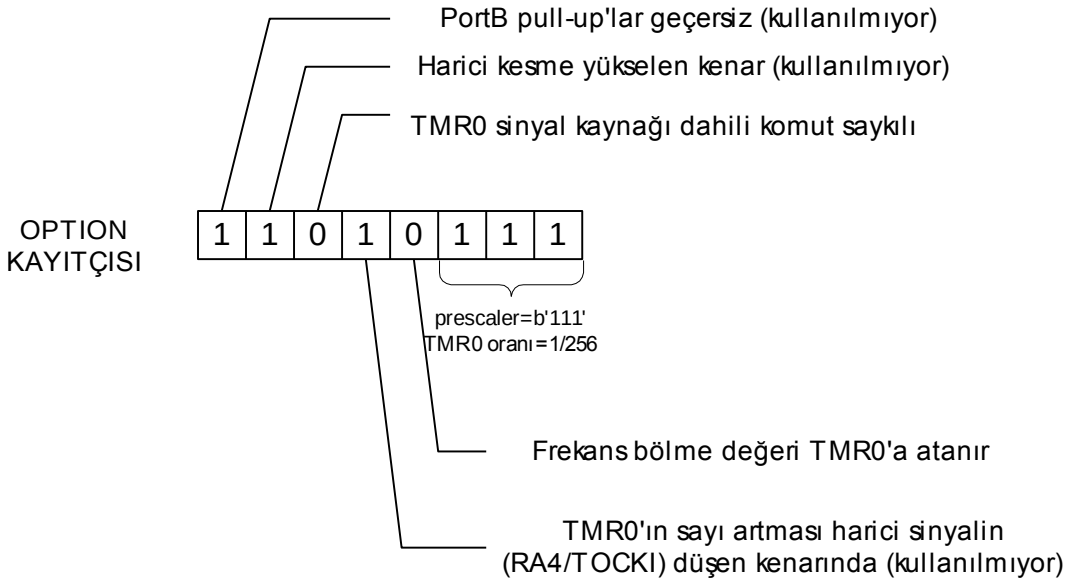
- 8 bitliktir.
- Herhangi bir anda sıfırlanabilir.
- Üzerine yazılabilir veya okunabilir.
- Programlanabilir frekans önbölücü değeri kullanılabilir.
- İçindeki veya dışındaki devrede bulunan osilatör saatleri kullanabilir.
- Dış sinyallerle düşen veya yükselen kenar tetiklenmesini yapabilir.
- Sayacı hep arttırarak sayma işlemi yapar.
- Timer0 ana program veya kesme alt programı çalışırken sayıcısını durdurmaz
- Uyuma modunda kullanılmaz
- Timer0 sayarak h 'FF' e geldiğinde, INTCON kesme yazmacının 2. biti uyarı bayrağını b '1' yapar. Bu uyarı biti kontrol edilerek zaman aşımı olup olmadığı anlaşılır.

Programda herhangi bir kesme oluştuğunda , o anda çalışmakta olan komut işlenir ve H'004' adresine sapma gerçekleşir. H '004' adresi kesme adresi olarak tanımlanır. Bu adreste bulunan komut kesme alt programını çalıştıracak olan GOTO komutudur. PIC kesme alt programının sonunda RETFIE komutuna geldiği zaman ana programda son işlenen komutun adresi yığından çıkarılır. Çıkarılan bu adres program sayacına yüklenir. Bundan sonra program sayacı normal çalışmasına devam ederek değerini bir arttırarak ana program komutlarını çalıştırmaya devam eder.

Örnek: PortB'nin 0. bitine bağlı LED'i flash yaptıran program.



```
        LIST        PIC16F877
        INCLUDE     "pic16f877.inc"
        BSF         STATUS,5    ; Bank1'e geç
        CLRF        TRISB       ; PORTB'nin tüm uçları çıkış
BASLA
        CLRWDT                  ; Prescaler atama işlemini hazırla
        MOVLW       b '11010111'  ; TMR0'ı yeni prescaler değerini ve sinyal kaynağı seç
        MOVWF       OPTION_REG; OPTION registerine yaz
        BCF         STATUS,5    ; Bank0'a geç
        CLRF        PORTB       ; PORTB'nin tüm çıkışları temizle
YAK
        BSF         PORTB,0     ; LED'i yak
        CALL        GECIKME     ; GECIKME alt programını çağır
SONDUR
        BCF         PORTB,0     ; LED'i söndür
        CALL        GECIKME     ; GECIKME alt programını çağır
        GOTO        YAK         ; yakıp-söndürmeye devam et
GECIKME
        CLRF        TMR0        ; TMR0'ı h '00' dan saymaya başla
TEST_BIT
        BTFSS       TMR0,5      ; TMR0'ın 5. bitini test et
        GOTO        TEST_BIT    ; Hayır 5. biti tekrar test et.
        RETURN
        END
```



TMR0 kayıtçısının tamamı okunabilir bir kayıtçıdır. TMR0'ın 5. biti 1 olduğunda ulaşılan sayı 32 dir. Yani burada kullanılan sayıcı 0'dan 32'ye kadar saydırılmaktadır. TMR0 içerisindeki sayılar TMR0 oranı 1/256 olduğu için 256 komut saykılında bir defa artacaktır.

TMR0, 32 ye kadar sayacağından 32'ye kadar sayma süresi;

256µS x 32 =8192µS →8.2 msn (4 MHz kristal osilatör kullanıldığı düşünülürse)