T.C.
FIRAT ÜNİVERSİTESİ
TEKNİK EĞİTİM FAKÜLTESİ
ELEKTRONİK VE BİLGİSAYAR EĞİTİMİ

BİL386 MİKROBİLGİSAYARLI SİSTEM TASARIMI

(Ders Notları)

İbrahim TÜRKOĞLU

ELAZIĞ – 2010

İÇİNDEKİLER

BÖLÜM 7: MPLAB

BÖLÜM 8: PIC PROGRAMLAYICI

BÖLÜM 9: PIC PROGRAMLAMA

BÖLÜM 10: UYGULAMALAR

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.

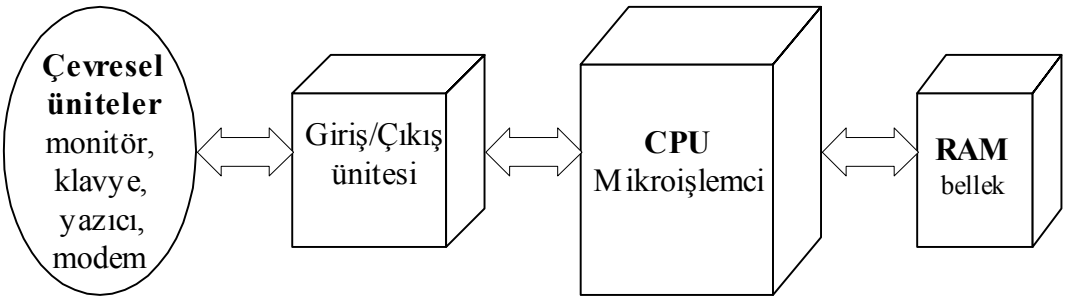
PIC mikrodenetleyicileri ise en yaygın ve en kullanışlı mikrodenetleyici olarak karşımıza çıkmaktadır.PIC mikrodenetleyicileri çok geniş bir ürün yelpazesine sahiptir. Her amaca yönelik bir mikrodenetleyici bulunabilmektedir. Bu mikrodenetleyicilerin en fazla kullanılmasının nedeni Flash Hafızaya sahip olmasıdır. Flash Hafıza teknolojisi ile üretilen bu bir belleğe yüklenen program, chip'e uygulanan enerji kesilse bile silinemez. Yine bu tip bir belleğe istenilirse yeniden yazılabilir. Bu belleğe sahip olan PIC leri programlayıp deneylerde kullandıktan sonra, silip yeniden program yazmak PIC ile yeni çalışmaya başlayanlar için büyük kolaylık sağlar.PIC mikrodenetleyicilerini programlamak için kullanılacak olan komutlar oldukça basit ve sayı olarak ta azdır. Bu nedenle tercih edilmektedirler.

Bu ders içersinde öğrencilere mikrodenetleyicili sistemlerin tasarım yöntemlerini, tasarım ölçütlerini, mikrodenetleyicilerin mimari farklılıklarını kavratmak ve çevre birimlerini programlamak ve yönetmek yeterlilikleri kazandırmak amacıyla geliştirilmiştir.

1.1. Mikroişlemciler ve Mikrodenetleyiciler

Mikroişlemci, 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 1.1 : Bir mikroişlemci sisteminin temel bileşenleri

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

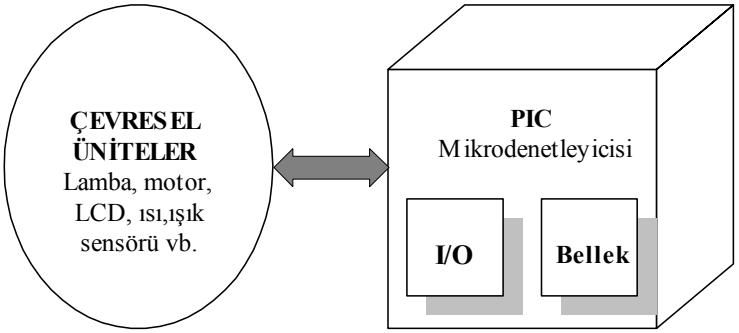
*CPU (**C**entral **P**rocessing **U**nit – Merkezi İşlem Birimi) :* Sistemin en temel işlevi ve organizatörüdür. Bilgisayarın beyni olarak adlandırılır.Komutları yürütmek, hesapları yapmak ve verileri koordine etmek için 4, 8, 16, 32 ve 64 bitlik sözcük uzunluklarında çalışır.

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.

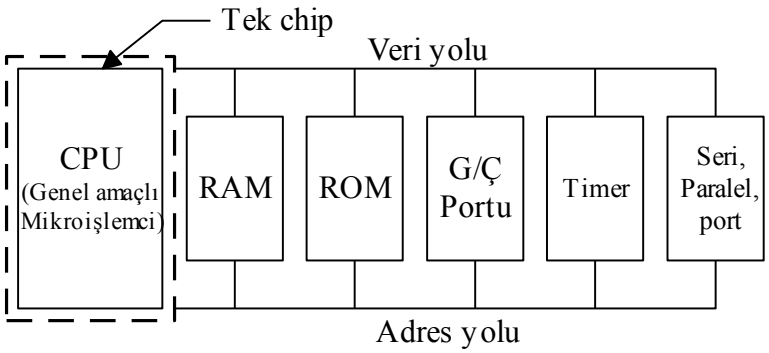
Diğer devreler : Mikroişlemci ile bağlantılı diğer devreler; sistemin kilitlenmesini önlemeye katkıda bulunan *Watchdog Timer,* mantık aşamalarını bozmadan birden fazla yonganın bir birine bağlanmasını sağlayan adres ve veri yolları (BUS) için *tampon (buffer),* aynı BUS 'a bağlanmış devrelerden birini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

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

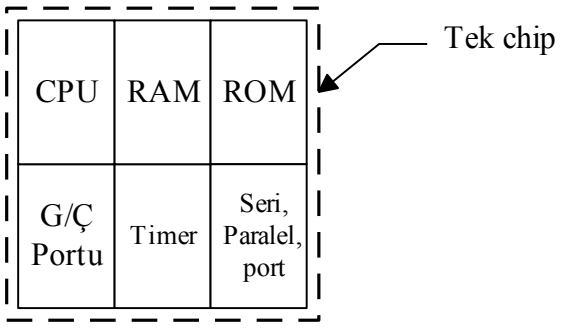


Şekil 1.2 : Bir mikrodenetleyici sisteminin temel bileşenleri

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



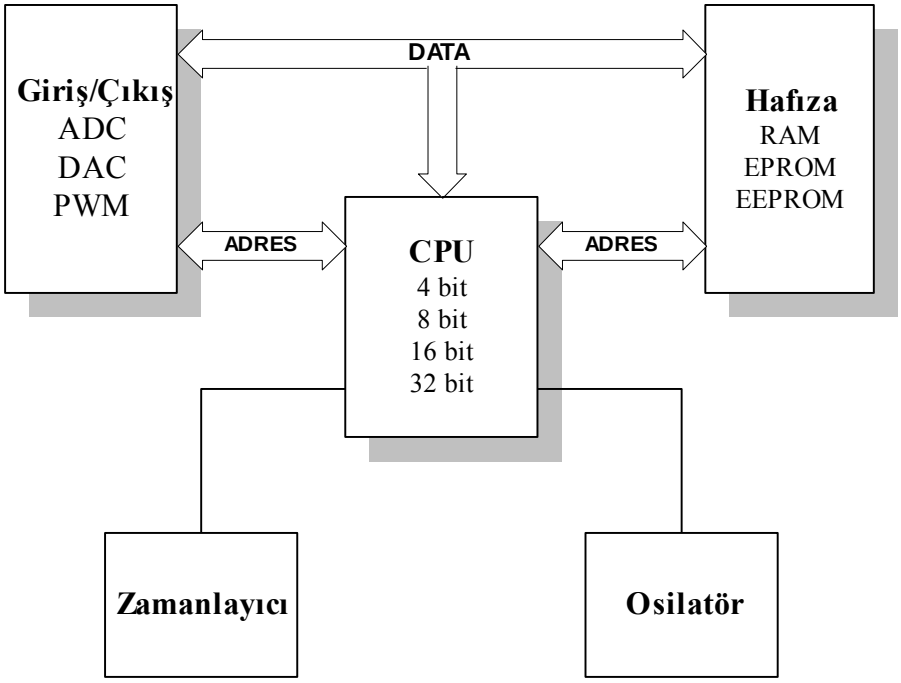
b) Mikrodenetleyici sistemi

Şekil 1.3 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 1990lı 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

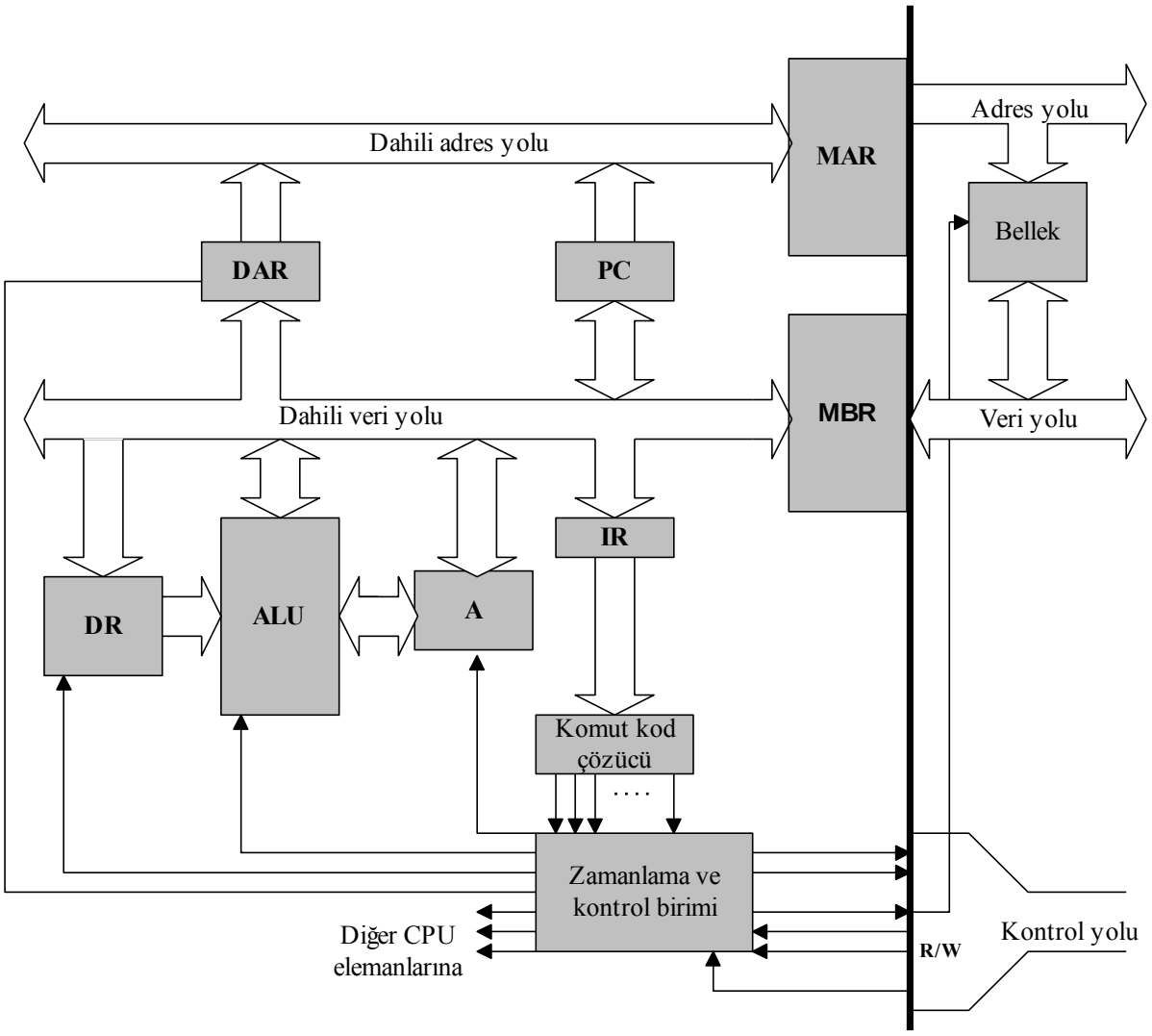


Şekil 1.4: Mikrodenetleyici sistem

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.

1.2. Basitten Karmaşığa Mikroişlemci Yapısı

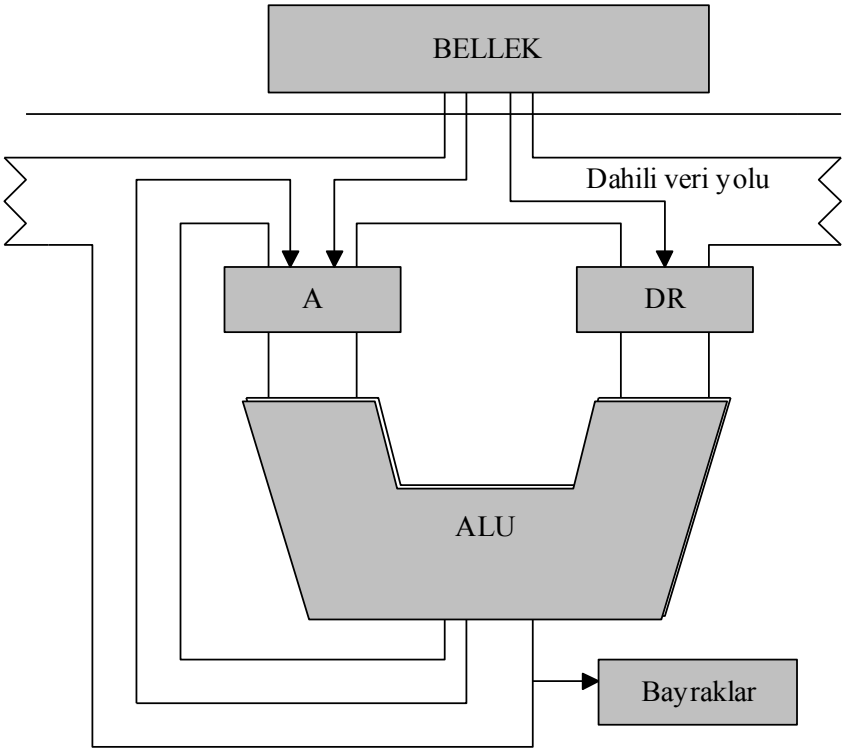
1.2.1. 8-Bitlik Mikroişlemciler: Basit bir işlemci kaydediciler, aritmetik-mantık birimi ve denetim birimi olmak üzere 3 ana bölümden meydana gelmiştir.



Şekil 1.5 : 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.



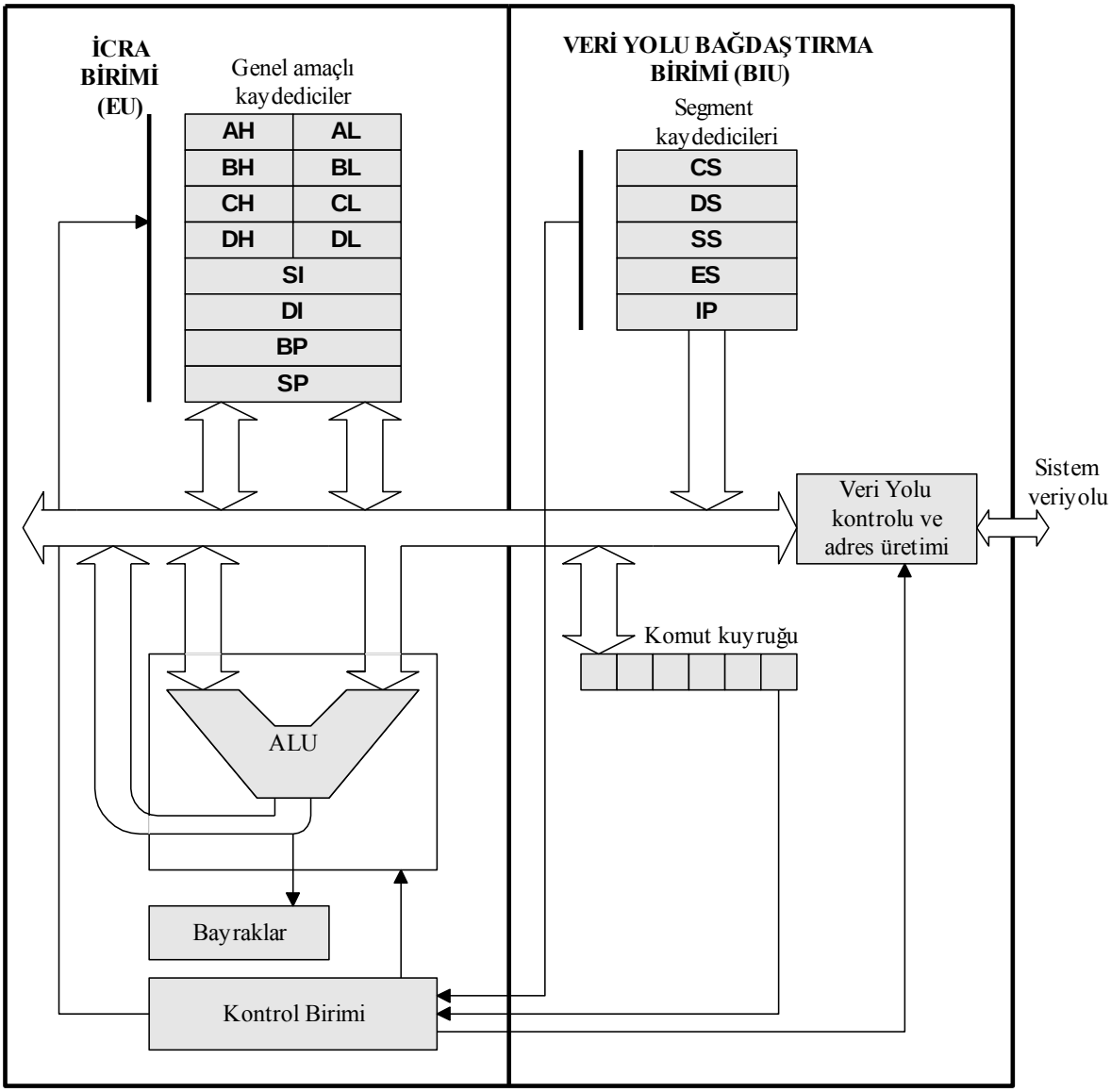
Şekil 1.6 : Aritmetik ve mantık birimi

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.

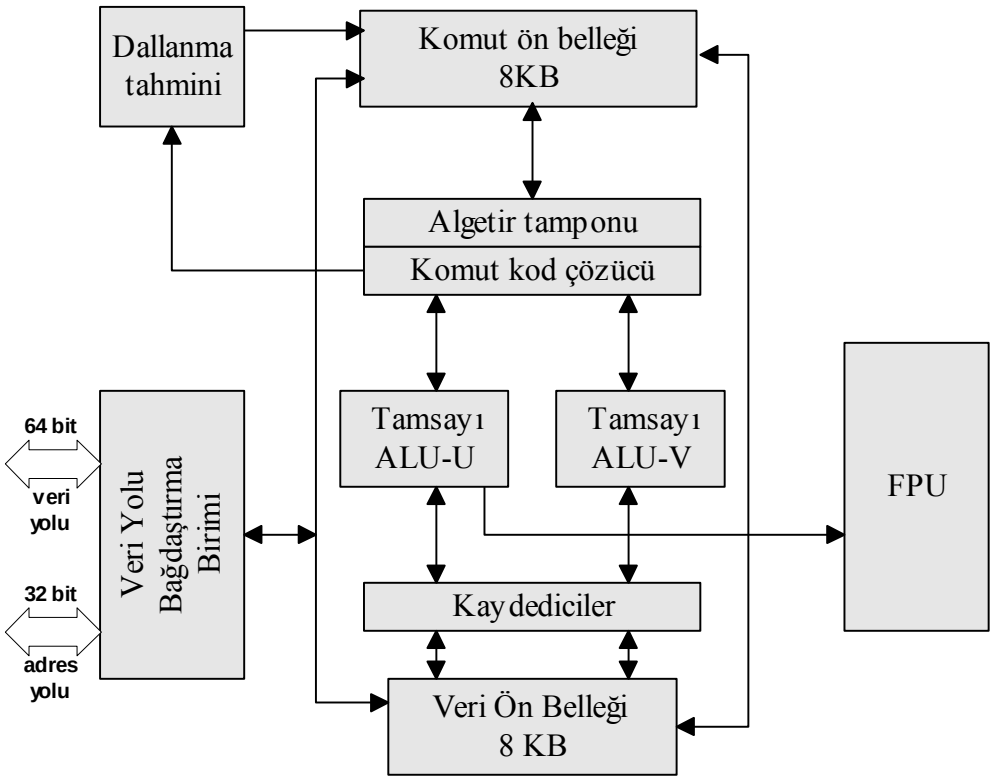
1.2.2. 16-Bitlik Mikroişlemciler: 16-bitlik mikroişlemciler basit olarak 8- bitlik mikroişlemcilerde olduğu gibi , Kaydediciler, ALU ve Zamanlama-Kontrol birimine sahiptir. Fakat mimari yapısı çoklu görev ortamına uygun hale getirildiğinden, işlemci içerisindeki bölümlerde fonksiyonel açıdan 2 mantıksal 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 1.7: 16- bitlik mikroişlemci mimarisi

1.2.3. 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 1.8 : 32-bitlik mikroişlemci mimarisi

1.3. Mikrodenetleyicilerin 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. Derlenen programda PICSTARTPLUS gibi çeşitli programlayıcılarla mikrodenetleyicilerin içerisine yüklenmektedir.

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

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.

1.4. 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.

1.5.Mikrodenetleyicilerin Yapısı

PIC mikrodenetleyicileri donanım olarak birbirlerine benzerler. Bazıları çok sayıda port ve zamanlayıcılara sahiptir ve PWM kanalları olanlarda vardır. Mikrodenetleyiciler de aritmetik işlemlerin sonucunun saklandığı yer STATUS kayıtçısı olarak adlandırılır. Ayrıca bu kaydedici bellek banklarının (bank 0, bank 1, bank 2 ve bank 3) seçimi içinde kullanılmaktadır. Mikroişlemcilerdeki akümülatörün (A,ACC) karşılığı PIC mikrodenetleyicilerinde W (Working Register-Çalışma Kaydedicisi) olarak verilmiştir. PIC mikrodenetleyicileri aritmetik ve mantık işlemleri için yalnızca bir ana kaydediciye sahip oldukları için diğer mikrodenetleyicilerden farklıdırlar. Sekiz bitlik W kaydedicisi CPU'dan veriyi başka bir yere transfer etmek için de kullanılır. Ayrıca WDT zamanlayıcısı da bulunmaktadır. PIC mikrodenetleyicileri içerisinde dahili osilatör devresi bulunmaktadır. PIC mikrodenetleyileri Harward mimarisi ile yapıldığı için oldukça hızlıdır. PIC mikrodenetleyicileri büyük oranda birbirine benzer donanıma sahiptirler. Herhangi birinin çalışması ve programlanması öğrenildiğinde rahatlıkla diğerleri de uygulamalarda kullanılabilmektedir.

1.6. 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" .*

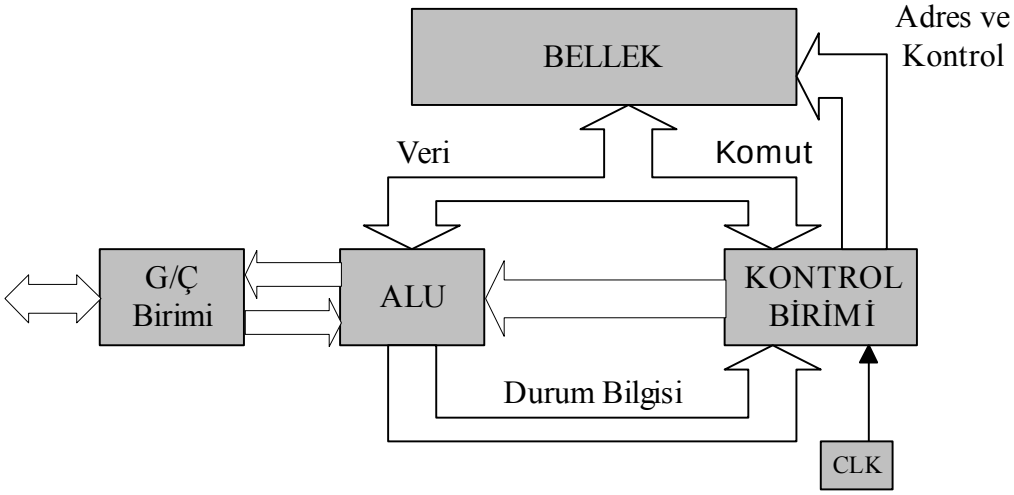
BÖLÜM 2. MİKRODENETLEYİCİ MİMARİLERİ

2.1. 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.

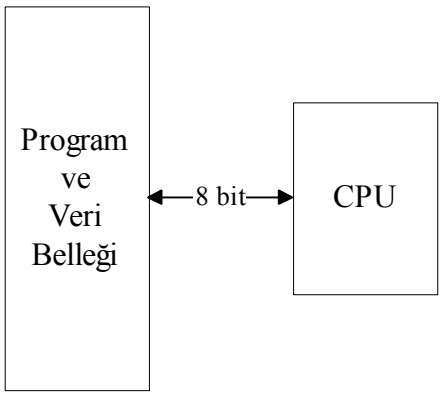
2.1.1. Von Neuman (Princeton) Mimarisi

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 2.1. 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 2.2. 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 2.1:

Mov acc, reg

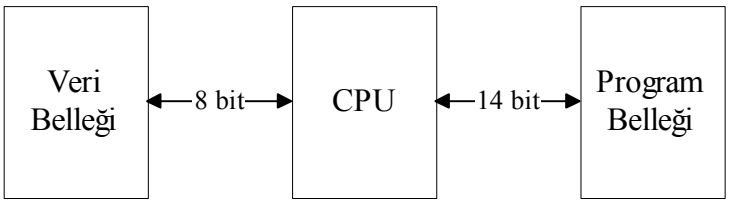
1. cp : Komut okur

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.

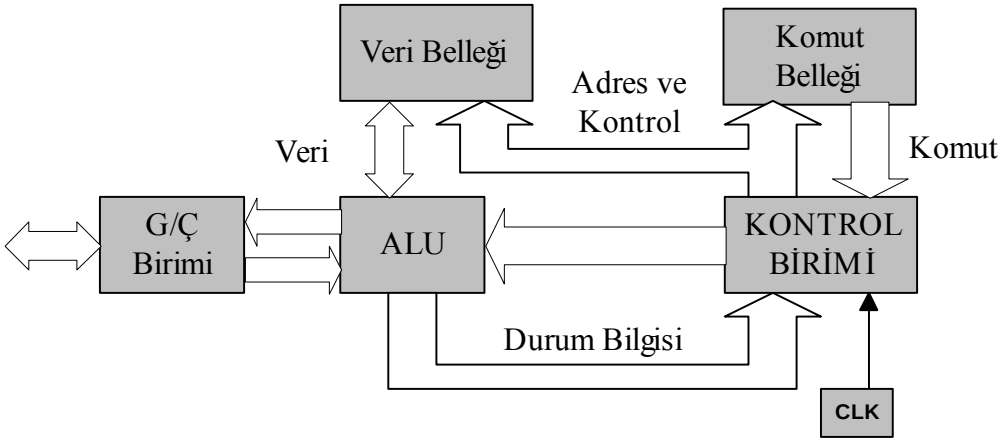
2.1.2. Harvard Mimarisi

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 2.3. 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 2.4. Harvard Mimarili bilgisayar sistemi

Örnek 2.2:

Mov acc, reg

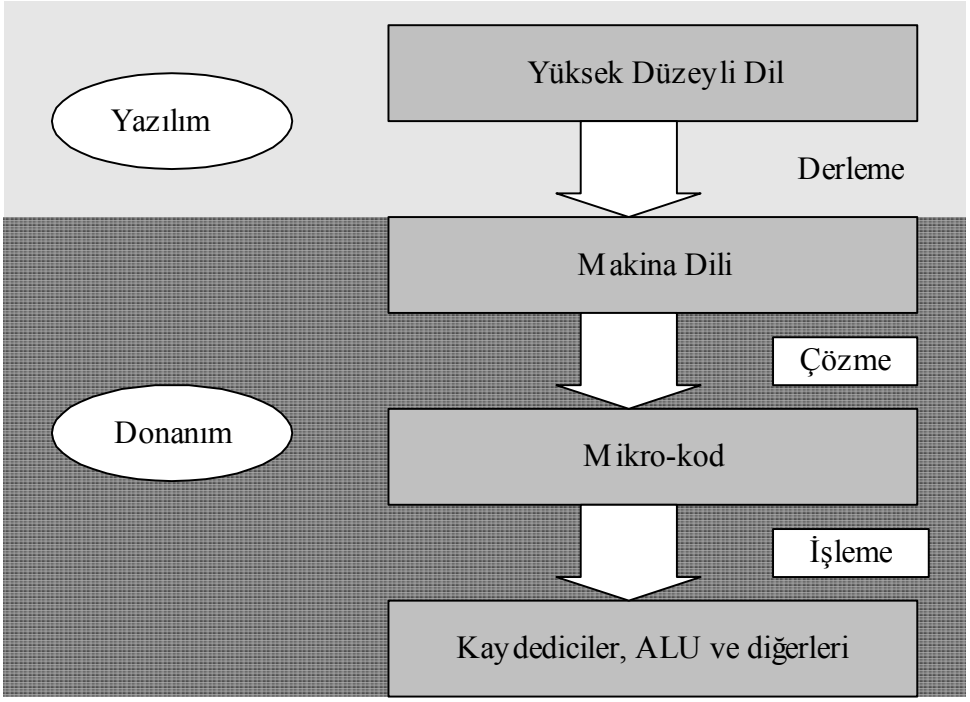
1. cp : Öncelikle *"move acc, reg"* komutunu okur.
2. cp : Sonra *"move acc, reg"* komutunu yürütür.

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.

2.2. Mikroişlemci Komut Tasarım Mimarileri

2.2.1 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 2.5. 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

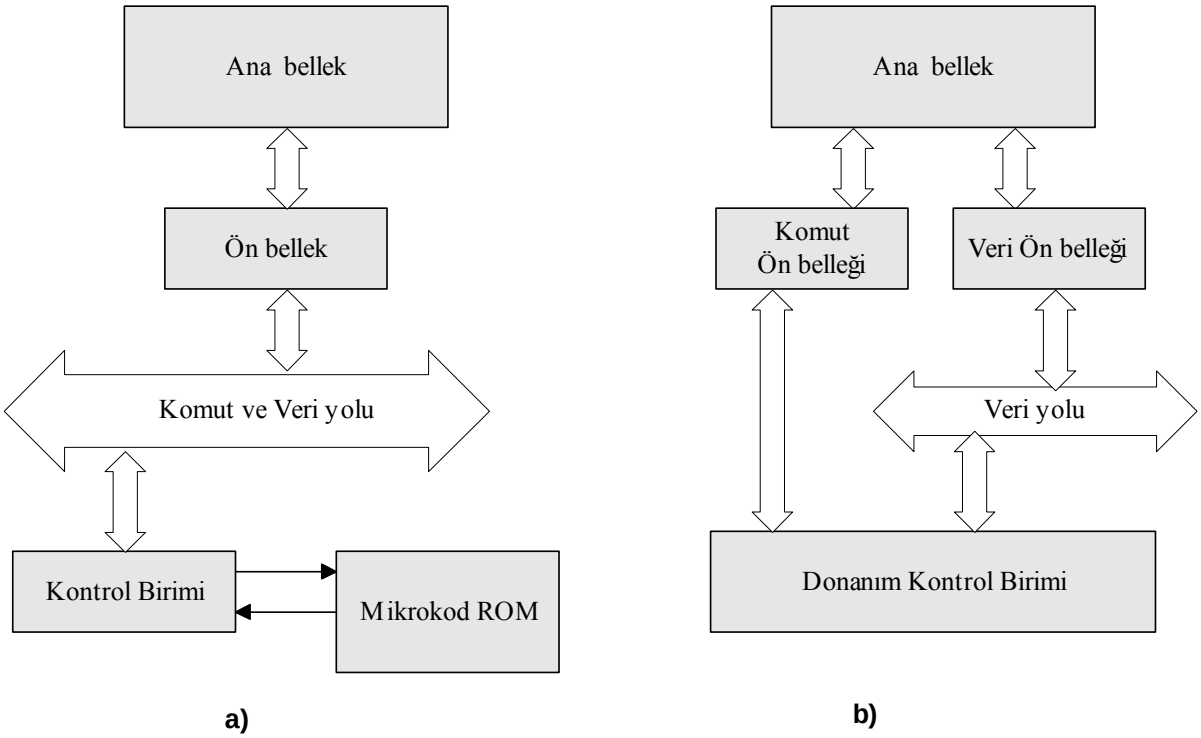
- Gelişen her mikroişlemci ile birlikte komut kodu ve yonga donanımı daha karmaşık bir hale gelmiştir.
- Her komutun çevirim süresi aynı değildir. Farklı komutlar farklı çevrim sürelerinde çalıştıkları için makinanın performansını düşürecektir.
- Bir program içerisinde mevcut komutların hepsi kullanılamaz.

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

2.2.2. 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 2.6. 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.

2.2.3. 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.

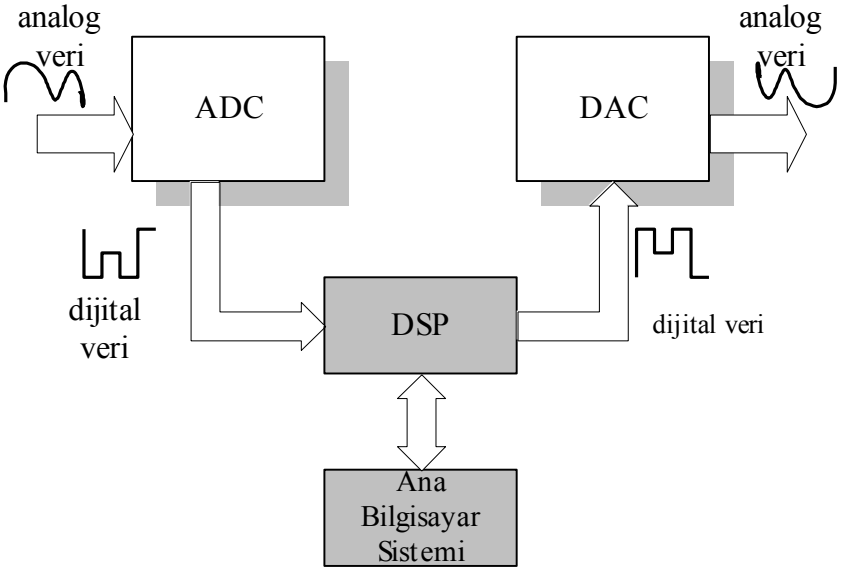
EPIC Mimarisin Üstünlükleri

- 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

2.2.4. 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 2.7. DSP sistem ve elemanları

2.3. 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 3. MİKRODENETLEYİCİLERİN 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 3.1. 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.

3.1. 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.

3.2. 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(=/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 3.2. 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 3.3. 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 3.1: 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 3.1: Programdaki komut sayısının I olduğunu varsayalım. Bu durumda;

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

=I × 2.0 × 10 ns = 20 I ns

CPU süresi B =I × 1.2 × 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 × I ns / 20 × I ns = 1.2

A makinesi B den 1.2 kat daha hızlıdır.

3.3. Yaygın Kullanılan Yanıltıcı Başarım Ölçütleri

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

3.3.1. 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

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

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

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

bu eşitliğe göre hızlı makinenin MIPS değeri yüksektir diyebiliriz.

3.3.1.1. 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.

ÖRNEK 3.2: Üç farklı tipte komutu olan makine düşünün, A tipi 1, B 2 ve C de 3 saat dönüşü tutsun. Makinenin saat hızı 100 MHz verilsin. Aynı programın iki farklı derleyiciden çıkmış kodlarının çalışma süresini ölçmeye çalıştığımızı düşünün;

Kodu oluşturan	**Komut sayısı (milyon)**		
	A-tipi	B-tipi	C-tipi
Derleyici 1	5	1	1
Derleyici 2	10	1	1

MIPS e göre hangi derleyicinin kod parçası daha hızlı çalışıyor?

Çalışma süreleri açısından hangi kod parçası daha hızlı çalışıyor?

Çözüm 3.2:

$$MIPS = \frac{Saat\ hızı}{CPI \times 10^6} = \frac{100\ Mhz}{CPI \times 10^6}$$

böylece

$$CPI = \frac{CPU\ saat\ dönüşü\ sayısı}{Komut\ sayısı}$$

burada

$$\text{CPU saat dönüşü sayısı} = \sum CPI_i \times \text{Komut sayısı}_i$$

her derleyici için toplam CPI yı bulmak amacıyla şu eşitliği kullanırız.

$$\text{CPI} = \frac{\sum \text{CPI}_i \times \text{Komut sayısı}_i}{Komut\ \text{sayısı}}$$

Durum 1	*Durum 2*
$CPI = \frac{((5\times1)+(1\times2)+(1\times3))\times10^6}{(5+1+1)\times10^6}$	$CPI = \frac{((10\times1)+(1\times2)+(1\times3))\times10^6}{(10+1+1)\times10^6}$
$= 10/7 = 1.43$	$= 15/12 = 1.25$
$MIPS_{derleyici-1} = \frac{100\times10^6}{1.43\times10^6} \cong 70$	$MIPS_{derleyici-2} = \frac{100\times10^6}{1.25\times10^6} \cong 80$

demek ki MIPS değerine göre derleyici2 daha yüksek başarımlıdır. Çalışma sürelerini hesaplarsak:

$$CPU\ süresi = \frac{Komut\ sayısı \times CPI}{Saat\ hızı}$$

$$CPU\ süresi\ 1 = \frac{(5+1+1)\times10^6 \times 1.43}{100\times10^6} = 0.10\ s$$

$$CPU\ süresi\ 2 = \frac{(10+1+1)\times10^6 \times 1.25}{100\times10^6} = 0.15\ s$$

çalışma sürelerine göre ise derleyici1 daha hızlı demektir. Demek ki MIPS değerine bakılarak varılan sonuç yanlıştır. Buda MIPS değeri, bilgisayarın başarımı için doğru bir ölçüt değildir anlamına gelir.

3.3.1.2.Tepe MIPS

İşlemcinin MIPS ölçümü hesabında kullanılan CPI'Yİ en aza indiren karışımı seçilerek elde edilir. Ancak, bu karışım tümüyle gerçekdışı ve uygulanamaz nitelikte olabilir. Bu yüzden tepe MIPS kullanışsız bir ölçüttür.

3.3.1.3. Göreceli MIPS

Göreceli MIPS şu şekilde hesaplanır.

$$Göreceli\ \ MIPS = \frac{T\text{ - bilinen} \times MIPS\ bilinen}{T\ ölçülen}$$

Burada

T-bilinen= Programın bilinen bir makinedeki çalışma zamanı

T-ölçülen= Programın ölçülecek makinedeki çalışma zamanı

MIPS- bilinen= Bilinen makinenin, genellikle VAX11/780, kabul görmüş MIPS değeri

Göreceli MIPS metriği sadece verilen bir program ve verilen girdi için doğrudur.

3.3.2.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.

3.3.2.1.Normalize MFLOPS

Normalize MFLOPS, yüksek seviye bir programlama dilindeki kayan noktalı işlemler için denk sayı bulma yöntemi tanımlar. Böylece bölme gibi daha karmaşık işlemlerle gerektiğince fazla ağırlık biçeriz. Bununla birlikte sayma/ağırlıklandırma farkı nedeniyle, normalize MFLOPS aslında kullanacağımız kayan noktalı işlemlerin gerçek sayısından çok farklı olabilir.

3.3.2.2.Tepe MFLOPS

Herhangi bir program parçası için mümkün olan en yüksek MFLOPS değerine tepe MFLOPS değeri denir. Başarım ölçmede tepe MIPS gibi tepe MFLOPS yanıltıcı bir ölçümdür.

3.3.3.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.

Bazı küçük programların çalışma süresinin çoğunu çok küçük bir kod parçasında geçirerek karşılaştırma program takımlarını yanılttığı tecrübeyle sabittir.Örneğin, SPEC karşılaştırma takımının ilk sürümündeki **matrix300 programı,** çalışma zamanın %99-unu tek bir komut satırında geçirir.Bundan yararlanan bazı şirketler matrix300 ün tek satırındaki başarımı arttırarak karşılaştırmayı yanıltmak üzere özel derleyiciler bile geliştirmiştir.

Küçük karşılaştırma programları elle bile hızla derlenebilir ve simüle edilebilir.Bunlar özellikle henüz derleyicisi yazılmamış yeni makinelerin tasarımları için kullanışlıdır.Ve de ,
Bunları standartlaştırmak kolay olduğundan, küçük karşılaştırma programlı başarı sonuçları yayınlanmış olarak kolayca bulunabilir.

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 3.4. Daha yüksek başarım sonucu elde edilen makine sisteminin betimlenmesi

3.3.4.Toplam Çalışma Zamanının Hesaplanması

Eğer iş yükündeki programlar eşit sayıda çalışırlarsa, karşılaştırma takımındaki n programın toplam yürütme süresi, her programın çalışma süresinin aritmetik ortalaması ile hesaplanır.

$$\text{Aritmetik Ortalama} = \frac{1}{\text{n}} \sum_{i=1}^{n} süre_i$$

burada iş yükünde n program vardır ve $süre_i$ i. programın yürütme süresidir.

Eğer iş yükündeki programlar farklı ağırlığa sahipse, her $süre_i$ terimini w_i ağırlığı ile çarpıp ağırlıklı aritmetik ortalama hesaplayabiliriz.

$$\text{Ağğırlıkl Aritmetik Ortalama} = \frac{1}{\sum_{\text{i=1}}^{\text{n}} \text{w}_\text{i}} \sum_{i=1}^{n} w_i \, \text{süre}_\text{i}$$

Normalize zamanın aritmetik ortalamasıyla hesaplanış toplam çalışma zamanı, özellikle programlardan birinin çalışma zamanı diğerlerinden çok yüksekse, gerçek başarımdan sapar. Normalize edilmiş zamanlar kullanılması durumunda, başarım geometrik ortalama kullanılarak daha iyi karşılaştırılabilir.

$$\frac{Geometrik\ Ortalama\ X}{Geometrik\ Ortalama\ Y} = Geometrik\ Ortalama(X/Y)$$ ve

$$\text{Geometrik Ortalama} = (\text{süre}_1 \times süre_2 \times ..... \times süre_n)^{(1/n)}$$

Geometrik ortalamanın aritmetik ortalamadan farkı birimsiz olmasıdır ve toplam yürütme süresiyle orantılı gitmez.Bu yüzdende programın yürütme süresini tahminde işe işe yaramaz.

İki programın, iki farklı makinedeki yürütme süreleri;

Karşılaştırma-takımı programları	**Yürütme süresi (saniye) Bilgisayar A**	**bilgisayar B**
Program 1	1	10
Program2	1000	100
Program3	1001	110

Normalize edilmiş aritmetik ortalama yanıltıcı olabilir.

Karşılaştırma programları	**Yürütme-süresi**		**A-ya normalize**		**B-ye normalize**	
	Ta	**Tb**	**Ta/Ta**	**Tb/Ta**	**Ta/Tb**	**Tb/Tb**
Program-1	1	10	1	10	0,1	1
Program-2	1000	100	1	0,1	10	1
Aritm.orta	500,5	55	1	5,05	5,05	1
Geom.orta	31,6	31,6	1	1	1	1

Veri A-ya normalize edildiğinde,B-nin başarımı A-nınkinin 5,05 katıdır, ama aynı veri B-ye normalize edildiğinde ,A-nın başarımı B-nin kinin 5,05 katıdır.geometrik ortalama iki durumda da tutarlıdır.

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.

3.4. Bölüm Kaynakları

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

BÖLÜM 4. 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.

4.1. 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.

4.2. PIC Program Belleği

PIC mikrodenetleyicileri üç tip bellek yapısı ile üretilmektedirler. Bunlar, ROM, EPROM ve FLASH bellek olarak adlandırılırlar. Flash bellek tipi yapısal olarak EEPROM'dan farklıdır. Flash bellek yapısı daha büyük miktarda veri saklamak için daha uygundur ve güç tüketimi daha azdır. Bu nedenlerden dolayı PIC'lerde program belleği Flash, veri belleği EEPROM yapıdadır.

ROM bellekli PIC mikrodenetleyicilerine üretim sırasında bir kez program yazılır ve yazılan program bir daha değiştirilemez. Yüksek miktarda seri üretimi yapılan elektronik sistemlerde ROM bellekli mikrodenetleyici kullanılması maliyet açısından avantaj sağlayabilir. Bu tip mikrodenetleyiciler CR kodu ile ifade edilir.(PIC12**CR**509A, PIC16**CR**56A)

EPROM bellekli PIC mikrodenetleyicilerinin üzerindeki program silinip yeniden yazılabilir. Bu tip mikrodenetleyicilerde yazılı programın siline bilmesi için kılıfın üzerindeki pencereden belli bir süre UV(Ultra-Violet) ışınına tutulmaları gerekir. Kılıfların üzerine pencere bırakılmayan EPROM bellekli mikrodenetleyicilerde ise silme işlemi yapılamaz. Bu tip mikrodenetleyiciler tek kez programlanabilirler (OTP-One Time Programmble). EPROM bellekli PIC mikrodenetleyiciler C kodu ile ifade edilir.(PIC12**C**509A, PIC16**C**56A)

Flash bellekli PIC mikrodenetleyicileri, program belleğine binlerce kez yazmaya olanak sağlarlar. Sadece programlayıcı devreleri veya ICSP (In Circiut Serial Programming- Devre Üzerinde seri Programlama) ile başka bir işlem yapmaya gerek kalmadan yeniden programlanabilirler. Bu açıdan, uygulama geliştirmede oldukça kullanışlıdırlar. Bu tip mikrodenetleyiciler F kodu ile ifade edilebilirler.(PIC112**F**629, PIC16**F**628, PIC16**F**877)

Uygulama geliştirirken kullanılacak olan PIC mikrodenetleyicisinin bellek yapısının yanı sıra bellek kapasitesinin de göz önünde bulundurulması gerekmektedir. PIC mikrodenetleyicilerinin program bellek kapasitesi 512 byte ile 64 kbyte arasında değişmektedir. Ayrıca RAM ve EEPROM veri belleği kapasitesine de dikkat edilmelidir.

4.3. Diğer Özelliklerine Göre PIC'ler

Bellek tip ve kapasitesinin yanı sıra, en yüksek çalışma frekansı da mikrodenetleyici seçiminde önemli bir etkendir. Bu seçim uygulamanın gereksinimi olan işlem hızı göz önüne alınarak dikkatle yapılmalıdır. PIC mikrodenetleyicileri tipine göre en fazla 40MHz frekansında

çalışabilir. Örneğin en fazla 4MHz çalışma hızını destekleyen bir PIC mikrodenetleyicisi için saat çevrimi (clock cycle) $1 / 4.10^6$ =250 ns'dir. PIC mikrodenetleyicisi komutları genellikle 4 saat çevriminde çalıştırır(Dallanma komutları hariç). Bu süreye komut çevrimi (instruction cycle) adı verilir. 250ns'liksaat çevrimi ile çalışan bir PIC'in komut çevrim süresi 250ns × 4 = 1µs'dir. Yazılacak program kodunun uzunluğu ve komut çevrim süresi dikkate alınarak mikrodenetleyici seçimi yapılmalıdır. PIC mikrodenetleyicilerinin en yüksek çalışma frekansı, tümdevrelerin üzerinde mikrodenetleyicinin model numarasında sonra 04/P, 10/P, 40/P şeklinde belirtilmektedir.

Uygun mikrodenetleyicinin seçiminde ele alınması gereken bir diğer özellik ise dış birimlerle mikrodenetleyicinin veri alışverişini sağlayan giriş/çıkış uçlarını sayısı ve tipidir. PIC mikrodenetleyicilerinde giriş/çıkış ucu sayısı 6 ile 68 arasında değişmektedir. PIC'lerde giriş/çıkış uçları ayrı ayrı giriş yada çıkış olarak programlanabilmektedirler.

Mikrodenetleyicilerin seçiminde uygulamanın gereksinimi olan özel donanımları içeren PIC' ler olabileceği göz önünde bulundurulmalıdır. Örneğin bazı PIC' ler içerisinde analog/sayısal dönüştürücü (ADC), sayısal/ analog dönüştürücü (DAC), gerçek zamanlı saat (RTC), darbe genişlik modülatörü (PWM) v.b. gibi donanımları içinde bulundurmakta ve diğer çevre birimlerle iletişimde kullanılan standart I²SPI, CAN, USB gibi haberleşme protokollerini donanım seviyesinde desteklemektedir.

4.4. 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.

4.5. PIC Donanım Özellikleri

4.5.1. 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 4.1. 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.

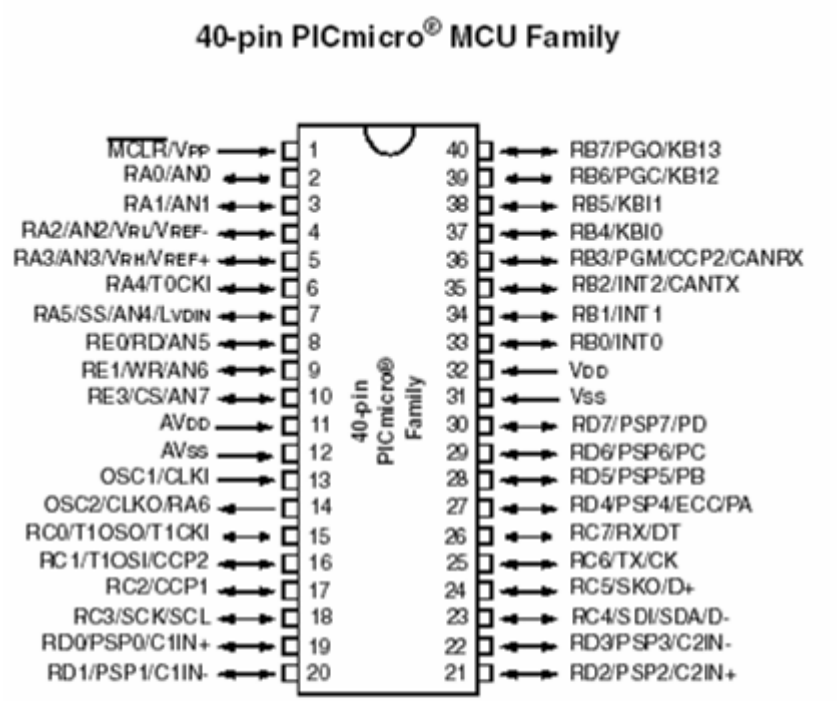
PIC programlayıcıları, program kodlarını yazarken bir komutun kaç bitlik bir sözcük uzunluğundan oluştuğuyla pek fazla ilgilenmezler. Seçilen bir yongayı programlarken uyulması gereken kuralları ve o yongayla ilgili özelliklerin bilinmesi yeterlidir. Bu özellikler; PIC 'in bellek miktarı, I/O portu sayısı, A/D dönüştürücüye sahip olup olmadığı, kesme fonksiyonlarının bulunup bulunmadığı, bellek tipinin ne olduğu (Flash, EPROM, EEPROM vb.) gibi bilgilerdir. Bu özellikleri aşağıdaki tabloda sunulmuştur:

Family		Architectural Features	Name	Technology	Products
PIC17CXXX	8-bit High-Performance MCU Family	• 16-bit wide instruction set • Internal/external vectored interrupts • DC - 25 MHz clock speed • 120 ns instruction cycle (@ 33 MHz) • Hardware multiply	PIC17C4X	OTP program memory, digital only	PIC17C42A, PIC17C43, PIC17C44
			PIC17CR4X	ROM program memory, digital only	PIC17CR42, PIC17CR43
			PIC17C75X	OTP program memory with mixed-signal functions	PIC17C756
PIC16CXXX	8-bit Mid-Range MCU Family	• 14-bit wide instruction set • Internal/external interrupts • DC - 20 MHz clock speed (Note 1) • 200 ns instruction cycle (@ 20 MHz)	PIC14CXXX	OTP program memory with A/D and D/A functions	PIC14C000
			PIC16C55X	OTP program memory, digital only	PIC16C554, PIC16C556, PIC16C558
			PIC16C6X	OTP program memory, digital only	PIC16C62, PIC16C62A, PIC16C63, PIC16C64, PIC16C64A, PIC16C65, PIC16C65A, PIC16C66, PIC16C67
			PIC16CR6X	ROM program memory, digital only	PIC16CR62, PIC16CR63, PIC16CR64, PIC16CR65
			PIC16C62X	OTP program memory with comparators	PIC16C620, PIC16C621, PIC16C622
			PIC16C7X	OTP program memory with analog functions (i.e. A/D)	PIC16C710, PIC16C71, PIC16C711, PIC16C715, PIC16C72, PIC16C73, PIC16C73A, PIC16C74, PIC16C74A, PIC16C76, PIC16C77
			PIC16F8X	Flash program and EEPROM data memory	PIC16C84 PIC16F83, PIC16F84
			PIC16CR8X	ROM program and EEPROM data memory	PIC16CR83, PIC16CR84
			PIC16C9XX	OTP program memory, LCD driver	PIC16C923, PIC16C924
PIC16C5X	8-bit Base-Line MCU Family	• 12-bit wide instruction set • DC - 20 MHz clock speed • 200 ns instruction cycle (@ 20 MHz)	PIC16C5X PIC16C5XA	OTP program memory, digital only	PIC16C52, PIC16C54, PIC16C54A, PIC16C55, PIC16C56, PIC16C57, PIC16C58A
			PIC16CR5X PIC16CR5XA	ROM program memory, digital only	PIC16CR54A, PIC16CR57B, PIC16CR58A
PIC12CXXX	8-bit, 8-pin MCU Family	• 12- or 14-bit wide instruction set • DC - 4 MHz clock speed • 1000 ns instruction cycle (@ 4 MHz)	PIC12C5XX	OTP program memory, digital only	PIC12C508, PIC12C509
			PIC12C67X	OTP program memory with analog functions	PIC12C671, PIC12C672

Note 1: The maximum clock speed for some devices is less than 20 MHz.

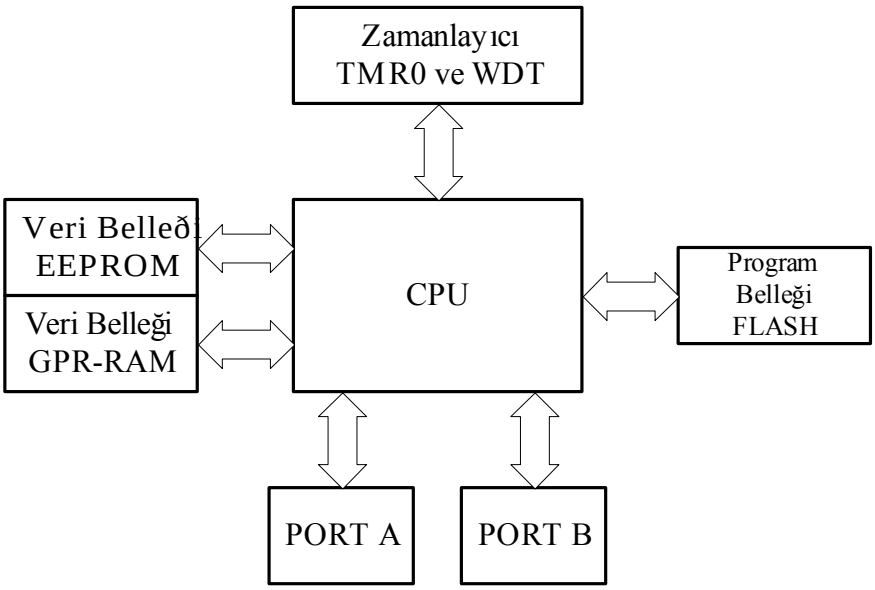
Tablo 4.2. PIC mikrodenetleyicilerinin genel özellikleri

4.5.2. PIC 'lerin Dış Yapıları

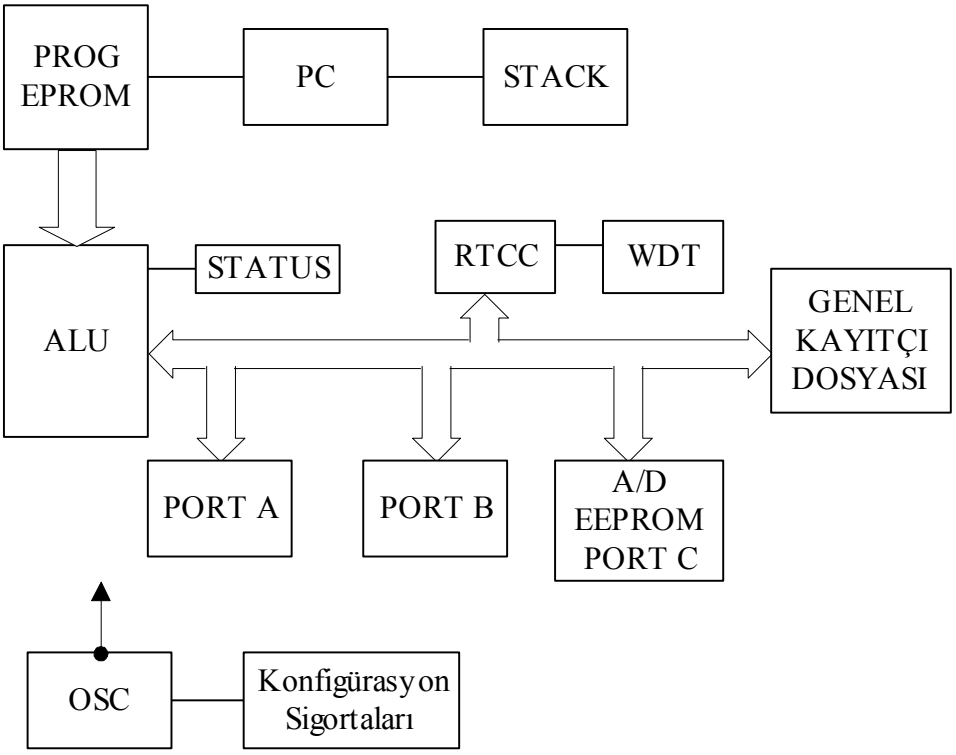


PIC16CR65	PIC16C765	PIC18F4220
PIC16C65B	PIC16C774	PIC18F4320
PIC16C67	PIC16F871	PIC18F4331
PIC16C662	PIC16F874	PIC18F4431
PIC16C74B	PIC16F874A	PIC18F442
PIC16C77	PIC16F877	PIC18F452
PIC16F74	PIC16F877A	PIC18F448
PIC16F77	PIC18C442	PIC18F458
	PIC18C452	

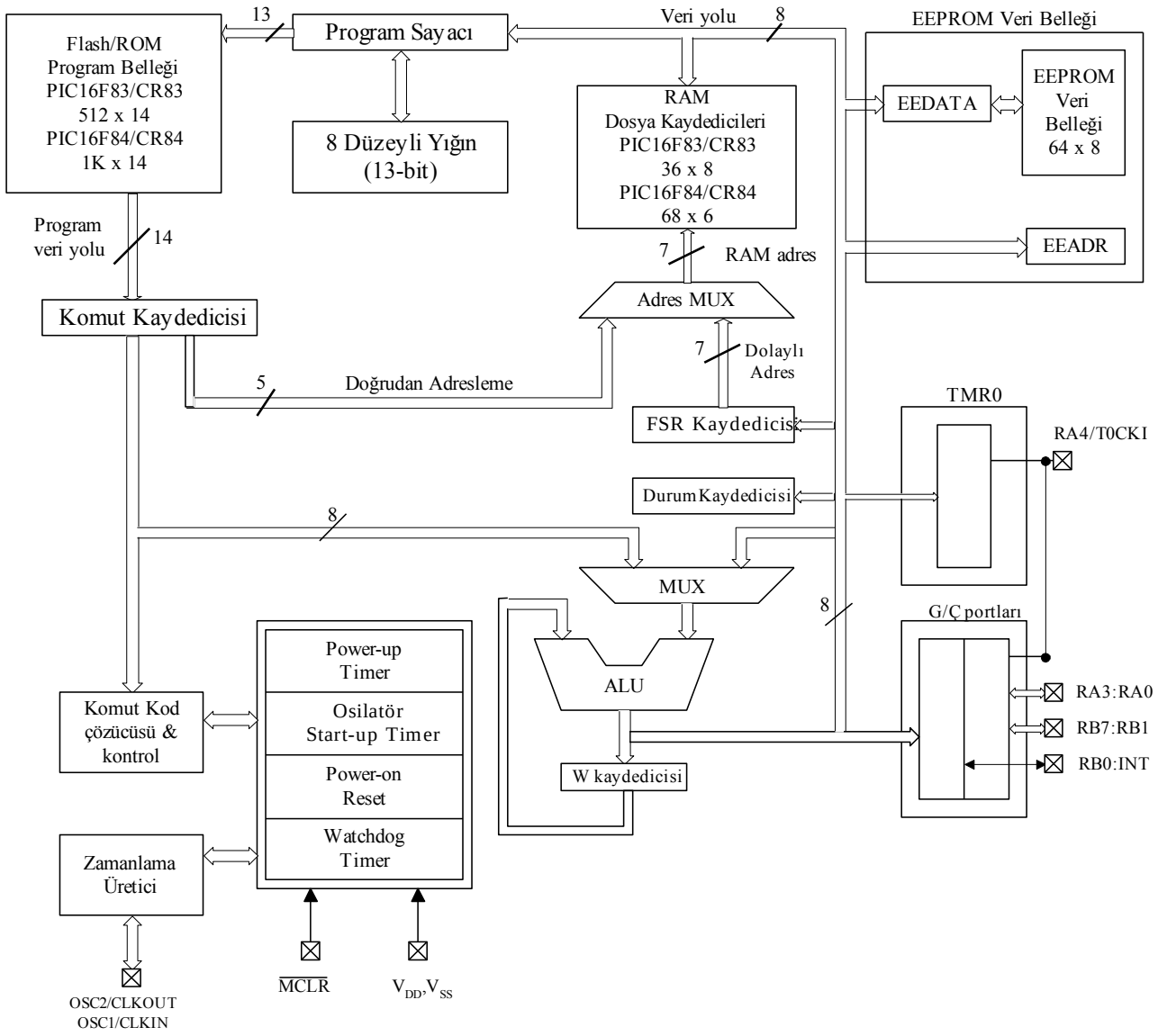
4.5.3. PIC İç Yapısı



Şekil 4.1. PIC 16F8X mikrodenetleyicilerinin genel blok diyagramı



Şekil 4.2. Temel PIC Blok Diyagramı



Şekil 4.3. PIC mikrodenetleyicilerinin genel blok diyagramı

4.5.4. 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.

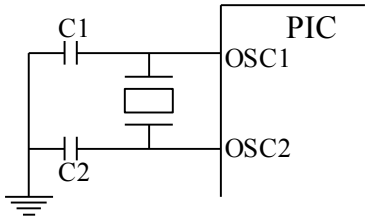
➢ ***Stack (Yığın) :*** PC, altprogram tamamlandığında, yani altprogramın bütün komutları çalıştırılıp bitince, altprogramın başlatılmasından hemen önceki adrese geri döner. Bunun için, altprogramın çalıştırılmasından bir önceki adres, yığın (stack) ismi verilen bir dizinin en üstüne konur (push). Bu 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 4.4. 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.

4.6. PIC Ürün Tanıma

Parça Numarası	**X**	**/XX**	**XXX**
Aygıt	Sıcaklık Aralığı	Paketlenme(kılıf)	Üretim Seri No

Aygıt PIC16F87X(1), PIC16F87XT(2), Çalışma değeri(V_{dd}) 4.0V-5.5V
PIC16LF87X(1), PIC16LF87XT(2), Çalışma değeri (Vdd) 2.0V-5.5V

Frekans Aralığı	04	=	4 MHz
	10	=	10 MHz
	20	=	20 MHz
Sıcaklık Aralığı	blank	=	0^oC - $+70^oC$ (Ticari)
	I	=	-40^{oC} - $+85^{oC}$ (Endüstriyel)
	E	=	-40^{oC} - $+125^{oC}$ (Genişletilmiş)
Kaplama	PQ	=	MQFP (Metric PQFP)
	PT	=	YQFP(Thin Quad Flatpack)
	SO	=	SOIC
	SP	=	Skinny plastic DIP
	P	=	PDIP
	L	=	PLCC

ÖRNEK 4.1:

a) PIC16F877 – 20/P 301= Ticari sıcaklık .

PDIP paket, 20 MHz, normal V_{DD} limitte, QTP örüntü=301.

b) PIC16LF876 – 04I/SO = Endüstriyel sıcaklık.

SOIC paket, 4 MHz, Genişletilmiş V_{DD} limit

c) PIC16F877 – 10E/P = Genişletilmiş sıcaklık.

PDIP paket, 10 MHz, normal V_{DD} limit

NOT:

1. F = CMOS FLASH
   LF = Low Power CMOS FLASH (Düşük güçlü CMOS FLASH)
2. T = Sadece SOIC, PLCC, MQFP, TQFP paketler

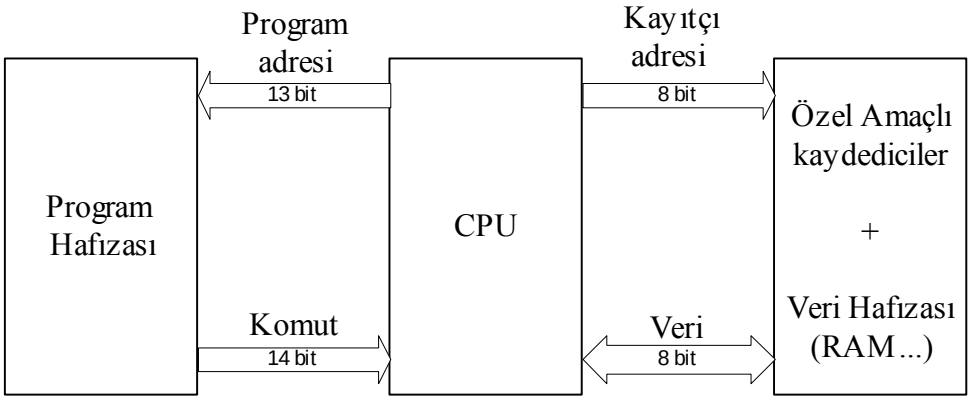
4.7. Bölüm Kaynakları

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.*

BÖLÜM 5. PIC16F877 MİKRODENETLEYİCİSİ

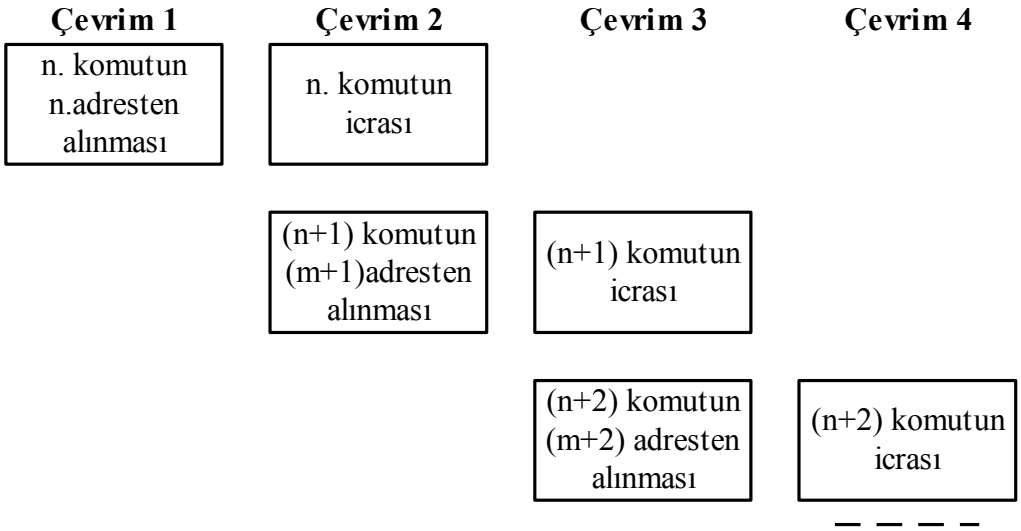
5.1. Genel Özellikleri

- 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.



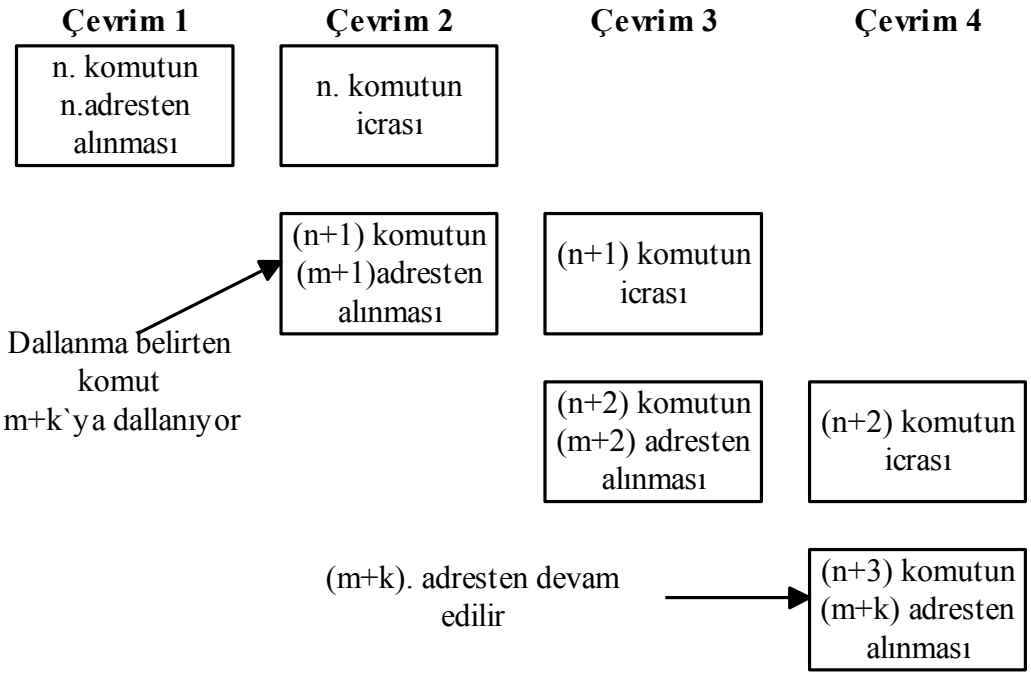
Şekil 5.1: PIC16F877 nin Harvard Mimarisi

- 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.



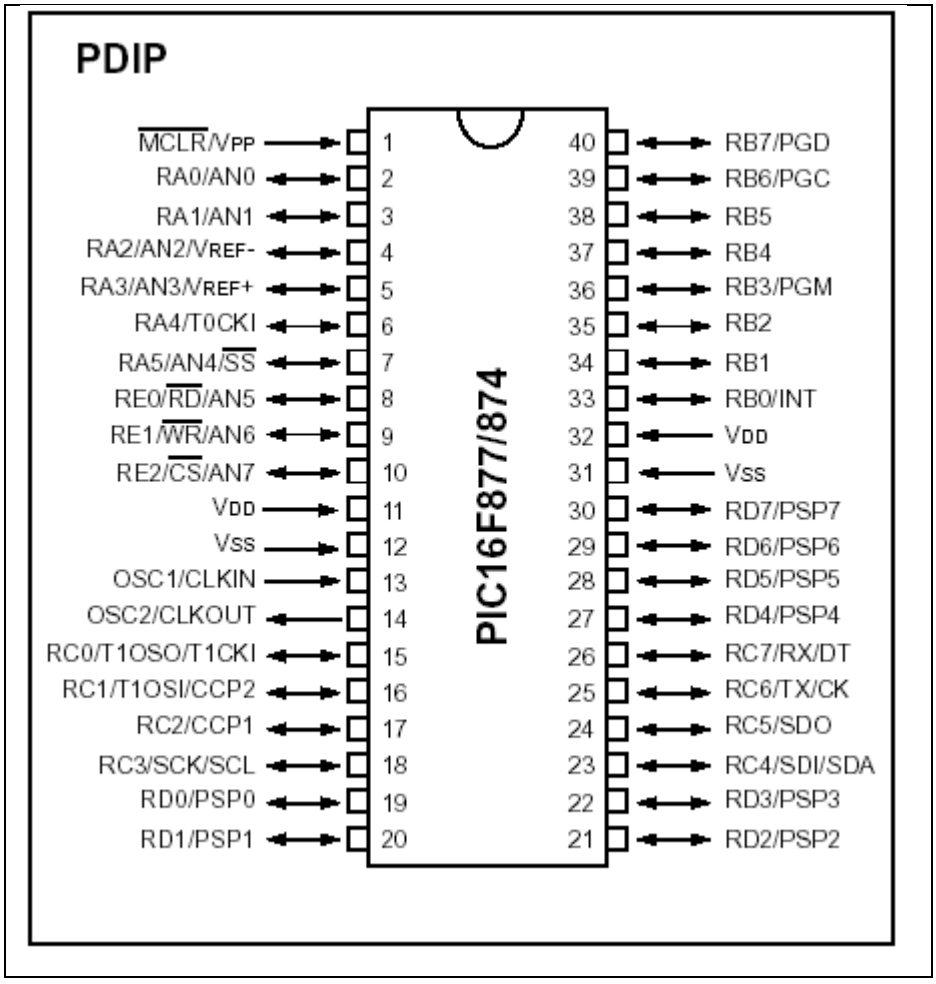
Şekil 5.2: Ardışıl adreslerden alınan komutların işhattı (pipelining)

- Dallanma yada sapma komutları ise, iki ardışık periyotta çalıştırılır ve program sayacı PC, iki arttırılır.



Şekil 5.3: Bir "goto" komutu için ilave saykılının gösterimi

- RISC mimarisine göre tasarlanmışlardır.
- Dış mimarisi : 40 pin



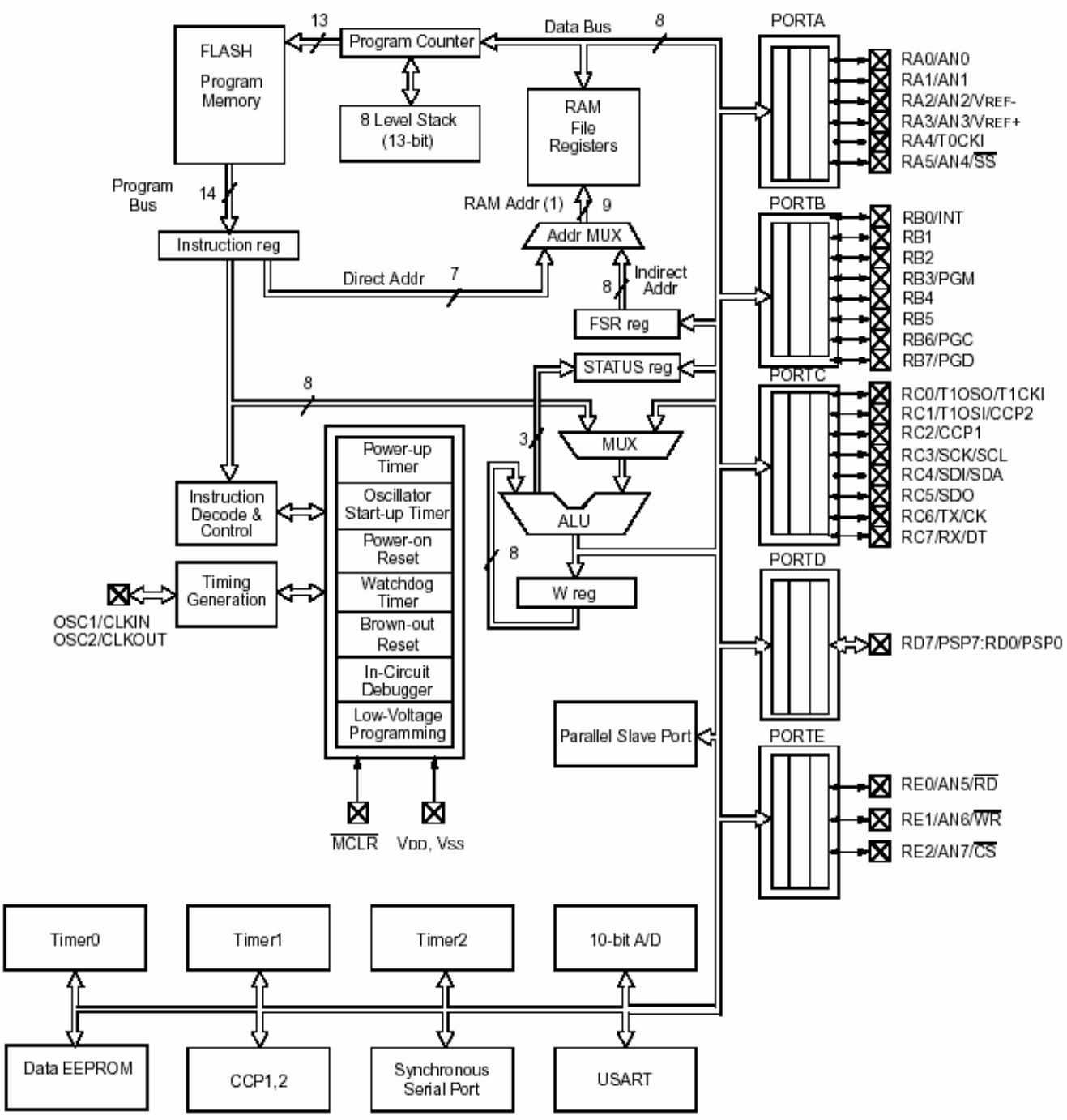
Şekil 5.4 : PIC16F87X Dış Mimarisi

- 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

5.2. İç Mimarisi

Aşağıdaki şekilde PIC16F877 yongasının temel donanım yapısı görülmektedir.

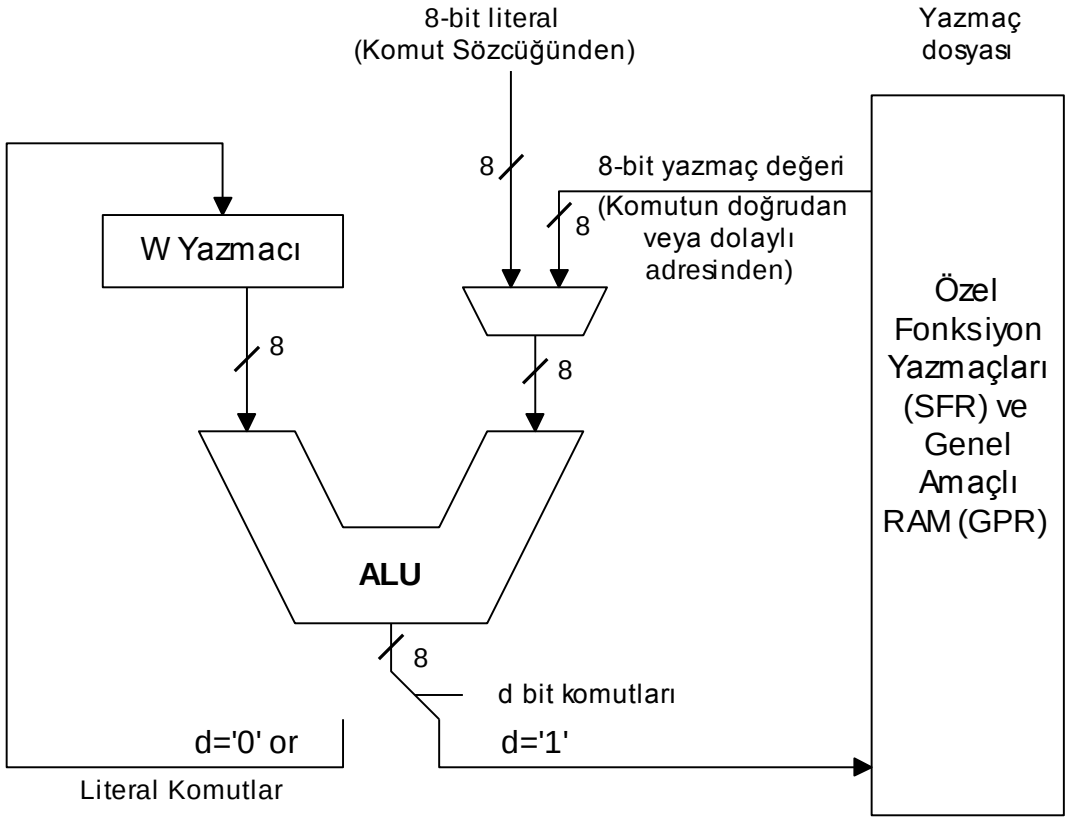


Şekil 5.5: PIC16F87X İç Mimarisinin Blok diyagramı

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 5.6. 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.

5.3. 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

5.4. 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

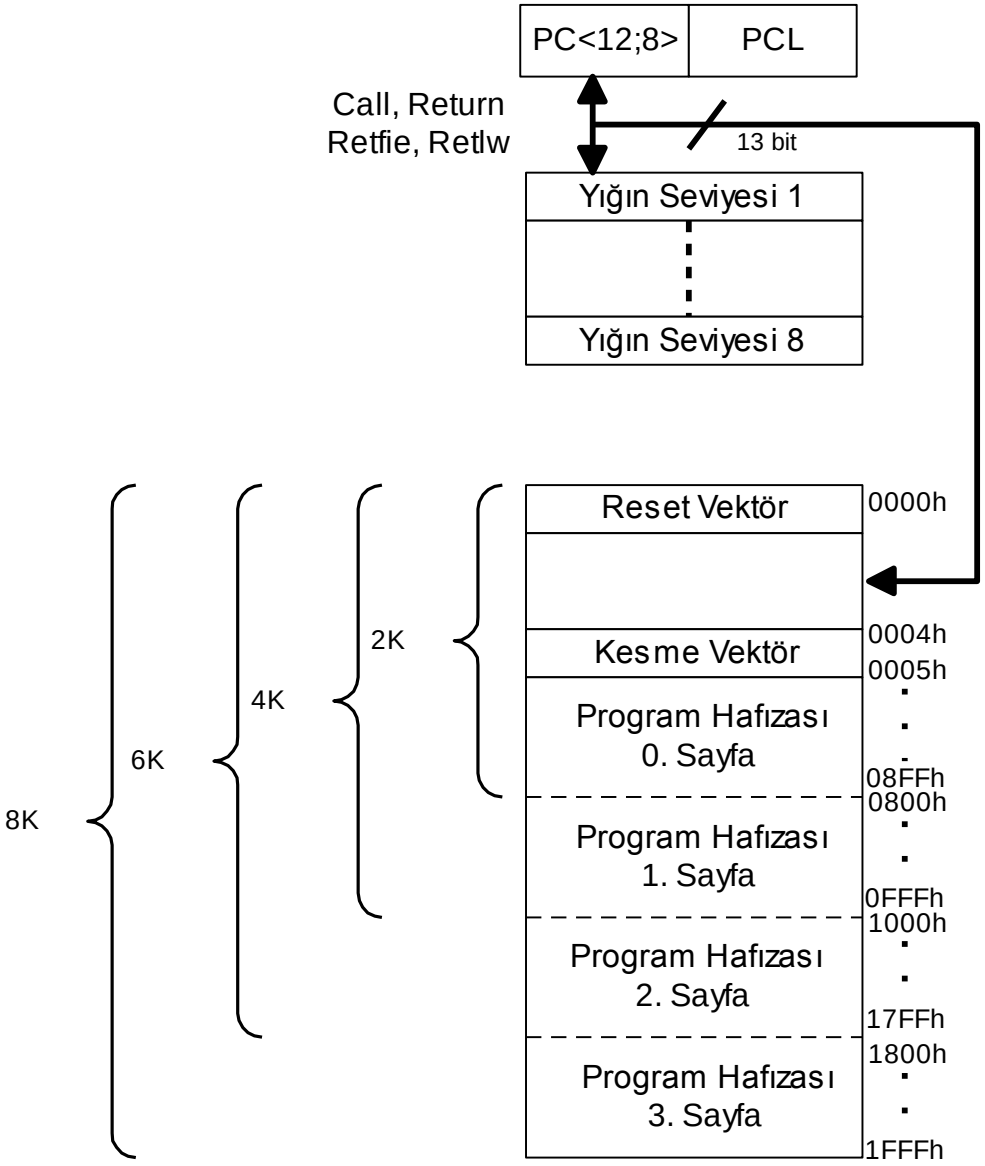
5.5. PIC Hafıza Organizasyonu

PIC16F877 mikrodenetleyicisinin hafıza yapısı iki ayrı bellek bloğundan oluşur:

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

5.5.1. 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 5.7: 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.

5.5.2. 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 5.8.: 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 5.9. Bellek Haritası

5.6. 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.

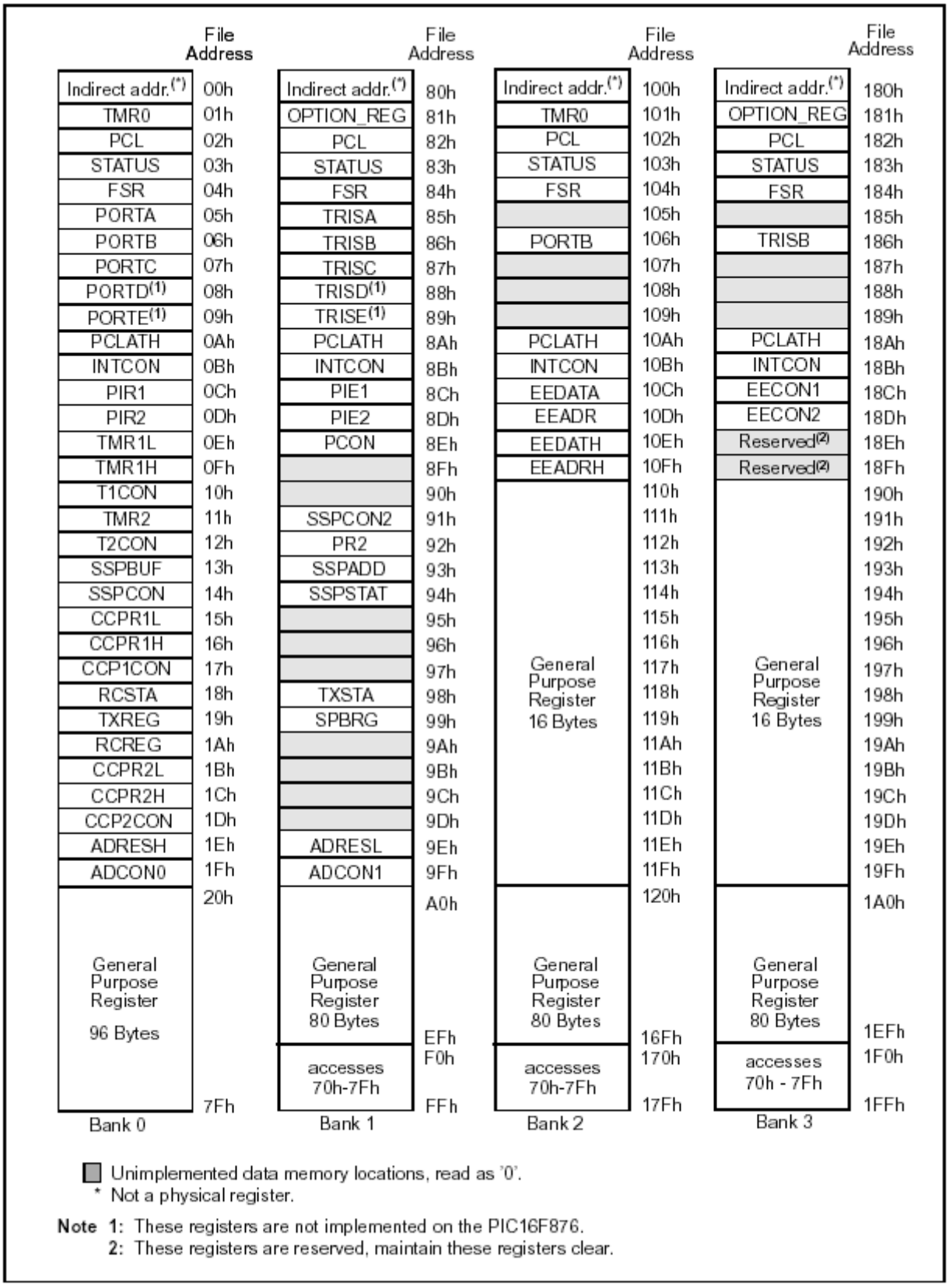
PCLATH: Program sayacının yüksek öncelikli byte yani, üst 5 biti için kullanılır.

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 5.10. 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.

5.6.1.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.

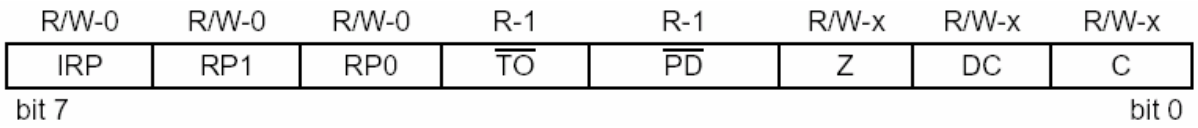
5.7. Durum (Status) Kayıtçısı

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 5.11: 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 / $\overline{\text{borç}}$ (carry / $\overline{\text{borrow}}$) 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 / $\overline{\text{borç}}$ 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.

5.8. 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 5.12: 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 5.13. Ön bölme seçme bitleri

5.9. 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 5.14. 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).

5.10. 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 5.15. 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.

5.11. 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: $0C_H$

R/W-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
bit 7							bit 0

Şekil 5.16. 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)

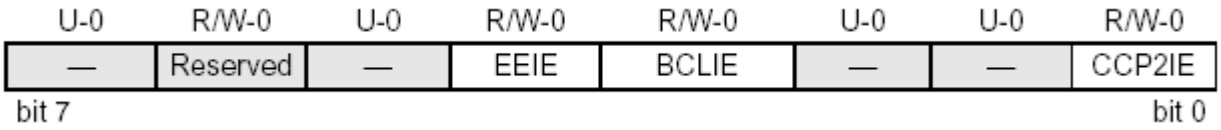
Bit 0 TMR1IF : TMR1 taşma kesmesi uyarı biti

0 = TMR1 kayıtçısı taşma olmadı.

1 = TMR1 kayıtçısında taşma oldu.(yazılımla temizlenmeli)

5.12. 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: $8D_H$



Şekil 5.17. 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.

5.13. 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 5.18. 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.

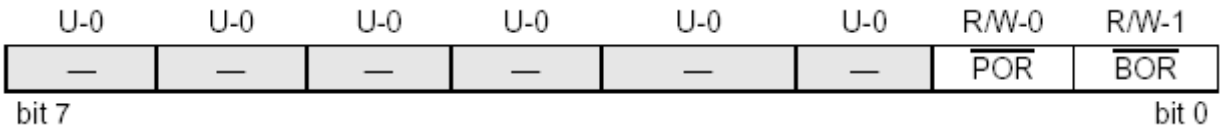
Bit 0 CCP2IF : CCP2 Kesme uyarı biti : Capture ve Compare modunda kullanılır. PWM modunda kullanılmaz.

0 = TMR1 kayıtçısı capture/compare vuku buldu.

1 = TMR1 kayıtçısı capture/compare vuku bulmadı.

5.14. 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 5.19: 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.

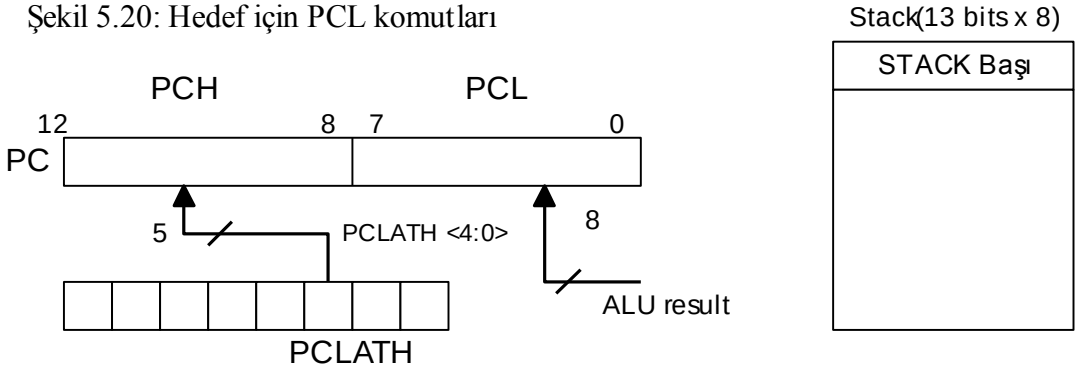
5.15. PCL ve PCLATH Adres Kayıtçıları

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.

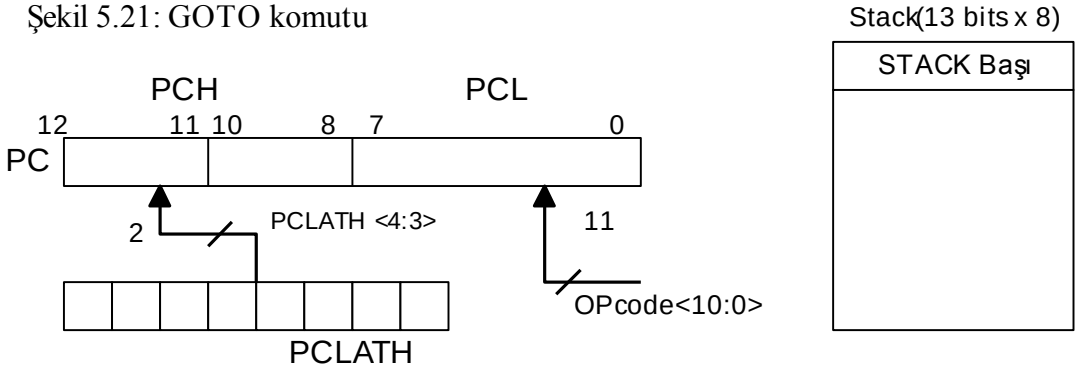
Bunlardan ilki PC yazmacının düşük öncelikli (low) byte ve yüksek öncelikli (high) byte'larının nereden ve nasıl yüklendiğini, ikincisi ise GOTO komutunda nasıl yüklendiğini açıklar. CALL komutunda ise PCL, PCH ve PCLATH ilişkisini gösterdiği gibi, yığınla PC ilişkisini de vurgulanmaktadır. CALL komutu,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

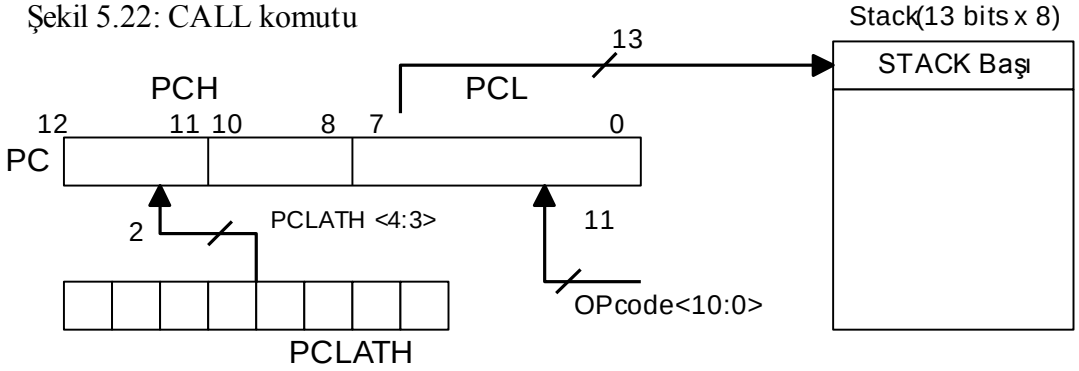
Şekil 5.20: Hedef için PCL komutları



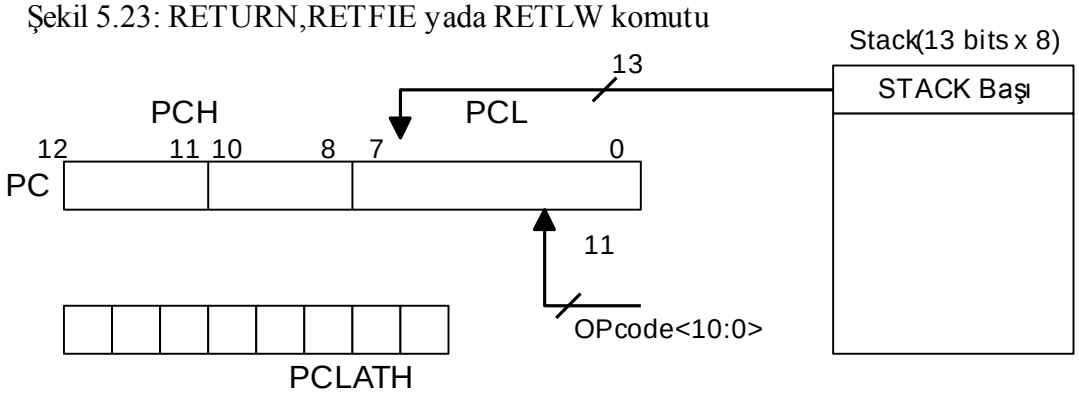
Şekil 5.21: GOTO komutu



Şekil 5.22: CALL komutu



Şekil 5.23: RETURN,RETFIE yada RETLW komutu



Şekil 5.24. 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).

5.16. 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.

5.17. 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)

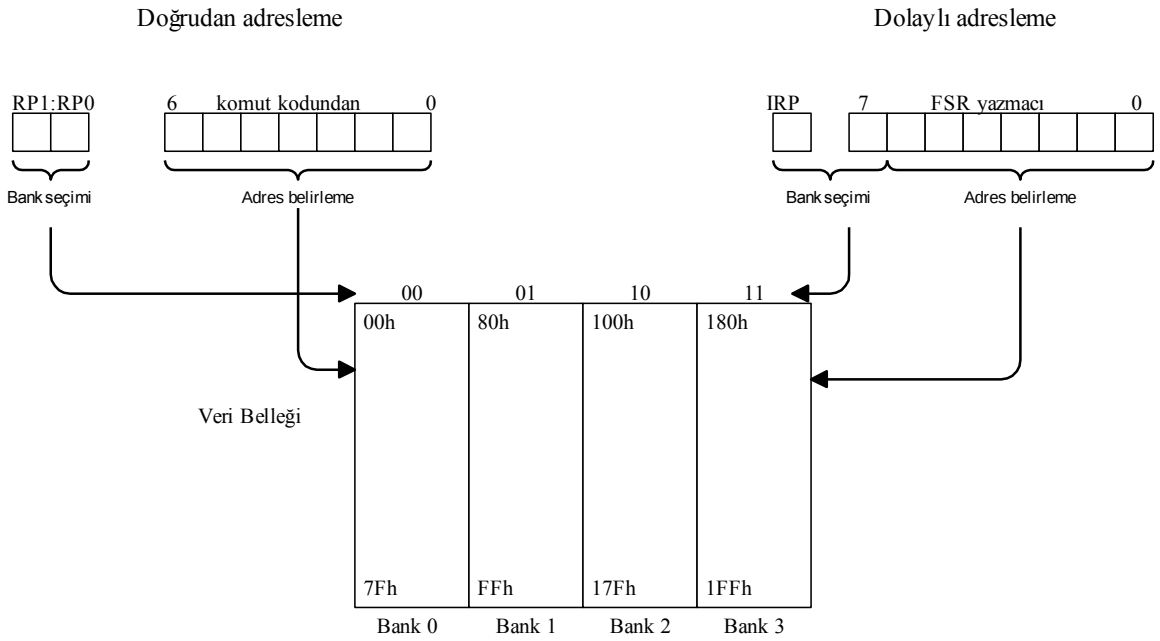
btfss FSR, 4; Hepsi yapıldı mı?

goto Sonraki; Temizlenecek alan bitmedi, sonrakine git.

Başka_Kısıma_Geç; Temizlenecek alan bitti.

….

Şekilde, doğrudan ve dolaylı adresleme nasıl oluşur gösterilmektedir.



Şekil 5.25. Doğrudan ve dolaylı adreslemede INDF ve FSR kullanımı

5.18. 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 5.26. 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.

5.19. 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*

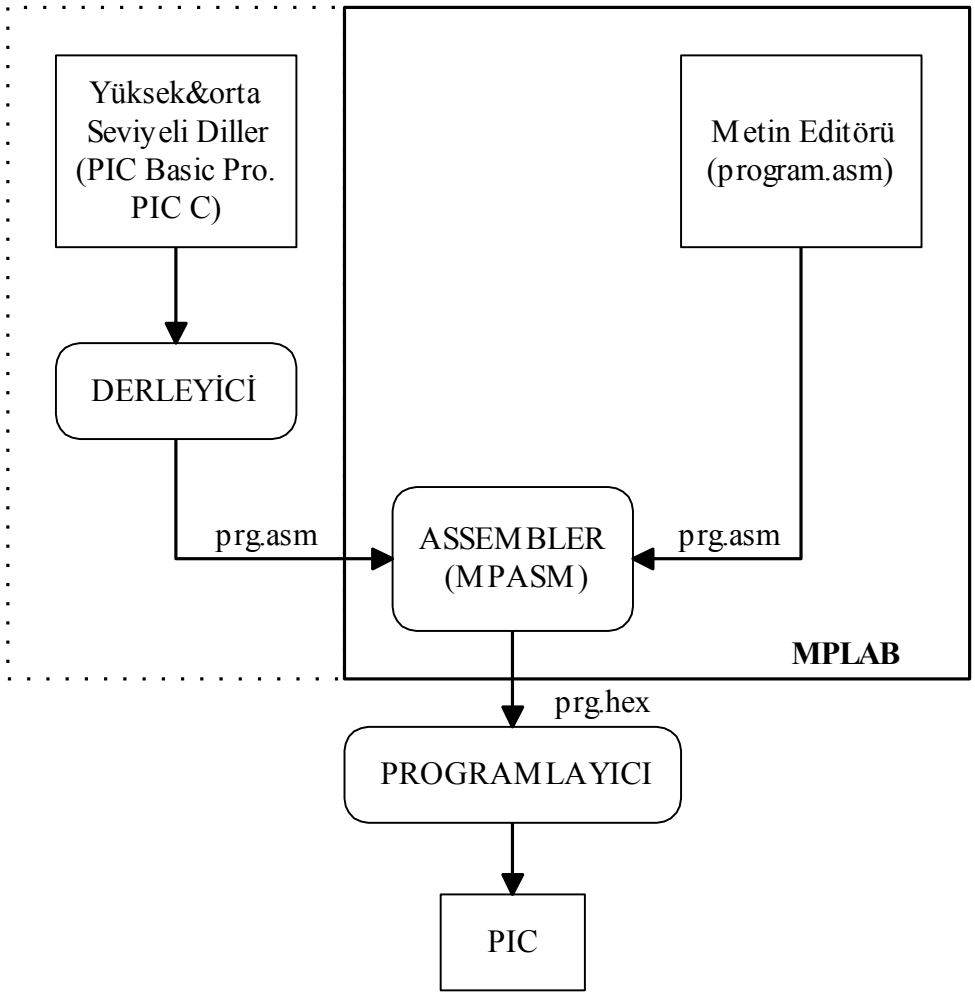
BÖLÜM 6. PIC PROGRAMLAMA VE ASSEMBLY DİLİ

6.1. 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 6.1 PIC programlama adımları

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)

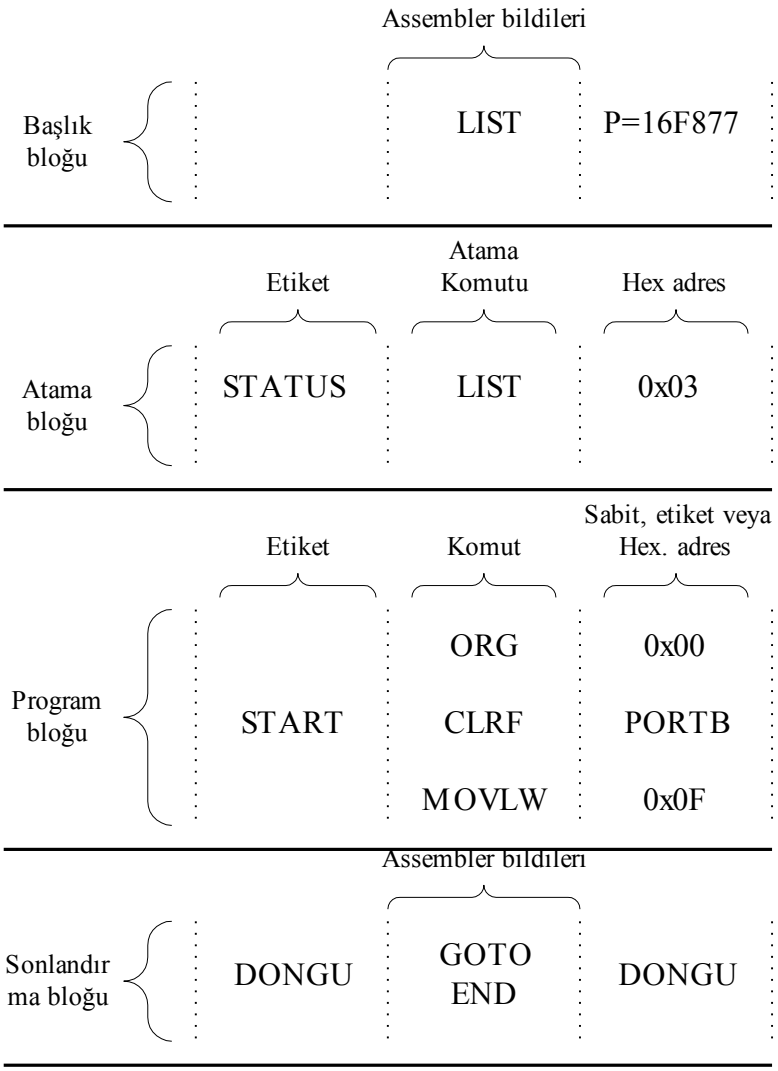
Etiketler PIC'in program ve veri belleğindeki belirli bir adresi isimlerle ifade etmeyi sağlar. Özellikle alt program çağrılmasında kullanılır. Assembler, programı derlerken etiketi gördüğü anda ilgili etiketin adresini otomatik olarak yerine koyup işlemi yapacaktır. Etiketler bir harfle veya "_" karakteri 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 6.2. Assembler komut kolonları

6.3. PIC Assembly Dilinde Sık Kullanılan İfadeler

6.3.1. # 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 6.1:

```
#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 6.2:

```
#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 6.3:

```
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 6.4:

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

SET

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

Örnek 6.5:

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

EQU

Program içerisinde sabit tanımlamamızı sağlayan ifadedir.

Örnek 6.6:

```
SAYAC       EQU   0x20
```

Bir etikete veri belleğinin bir adresini atamamızı sağlar.

ORG

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

Örnek 6.7:

```
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 6.8:

```
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 6.9:

```
……
……
end
```

WHILE, ENDW

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

Örnek 6.10:

```
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 6.11:

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

_CONFIG

Bildirimi takip eden açıklamaları işlemcinin konfigürasyon bitlerine aktarmayı sağlayan ifadedir.

Örnek 6.12:

```
_CONFIG h 'FFFF'
```

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.

6.4.1. 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.

6.4.2. PIC16F877 Komut Kümesi

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

Komut Yazılımı	Komut Tanımlaması	Çevrim	14-bitlik Opcode	STATUS

		Süresi	MSb LSb	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.

6.4.3. Kayıtçı Adresleme Modları

PIC lerde başlıca üç tip adresleme modu vardır:

6.4.3.1. Anında(Immediate) Adresleme:

Örnek 6.13: MOVLW H'0F' ; W = OF → Komut Sözcüğü : 11 0000 0000 1111 =300F

6.4.3.2. 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 6.14:

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

6.4.3.3. 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 6.15: 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.
```

6.4.4. PIC Assemblyde Sayıların ve Karakterlerin Yazımı

6.4.4.1 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'
```

6.4.4.2. Binary Sayılar

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

```
MOVLW     b '00001010'
```

6.4.4.3. 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'
```

6.4.4.4. ASCII Karakterler

ASCII karakterler yazılırken RETLW komutu ile yazılır.

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

6.4.5.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 6.16: 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 6.17: ADDWF f,0

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

Örnek 6.18: 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 6.19: 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 6.20: 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 6.21: 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 6.22 : 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 6.23: 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 6.24 :

```
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 6.25.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 6.25.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 6.25.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 6.26: 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 6.27: 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 6.28: 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 6.29: 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 6.30: 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 6.31: 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 6.32: 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

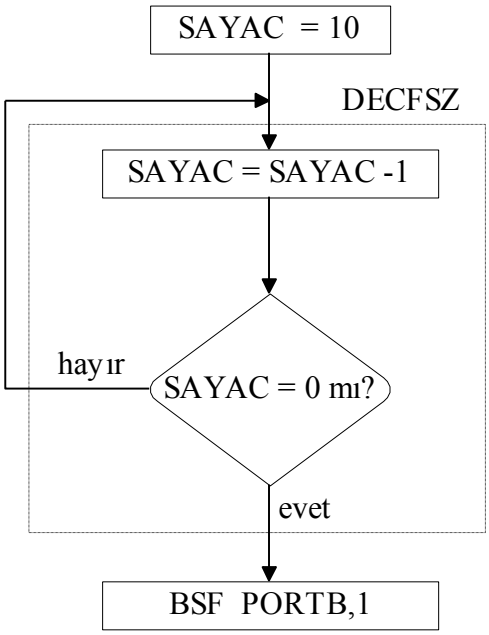
Status etkisi : Yok

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 6.33:

```
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 6.34: 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 6.35: 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 6.36:

```
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 6.37: 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 6.38: IORWF f,0

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

Örnek 6.39: 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 6.40: 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 6.41: MOVF f,0

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

Örnek 6.42: 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 6.43: 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 6.44: 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 6.45: 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 6.46: 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 6.47: RETURN

Bu komut uygulandıktan sonra PC' ye TOS değeri yerleştirilir.

28-) RLF Yazmaç bitlerini sola doğru döndür

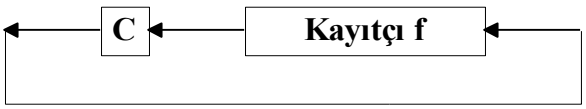
Söz dizim kuralı : [etiket] RLF f,d

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 6.48:

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

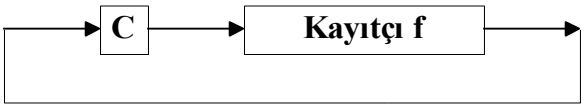
Söz dizim kuralı : [etiket] RRF f,d

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 6.49:

```
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 6.50:

```
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 6.51:

```
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 6.52:

```
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 6.53:

```
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 6.54: 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 6.55: XORWF f, 0

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

6.5. 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.*

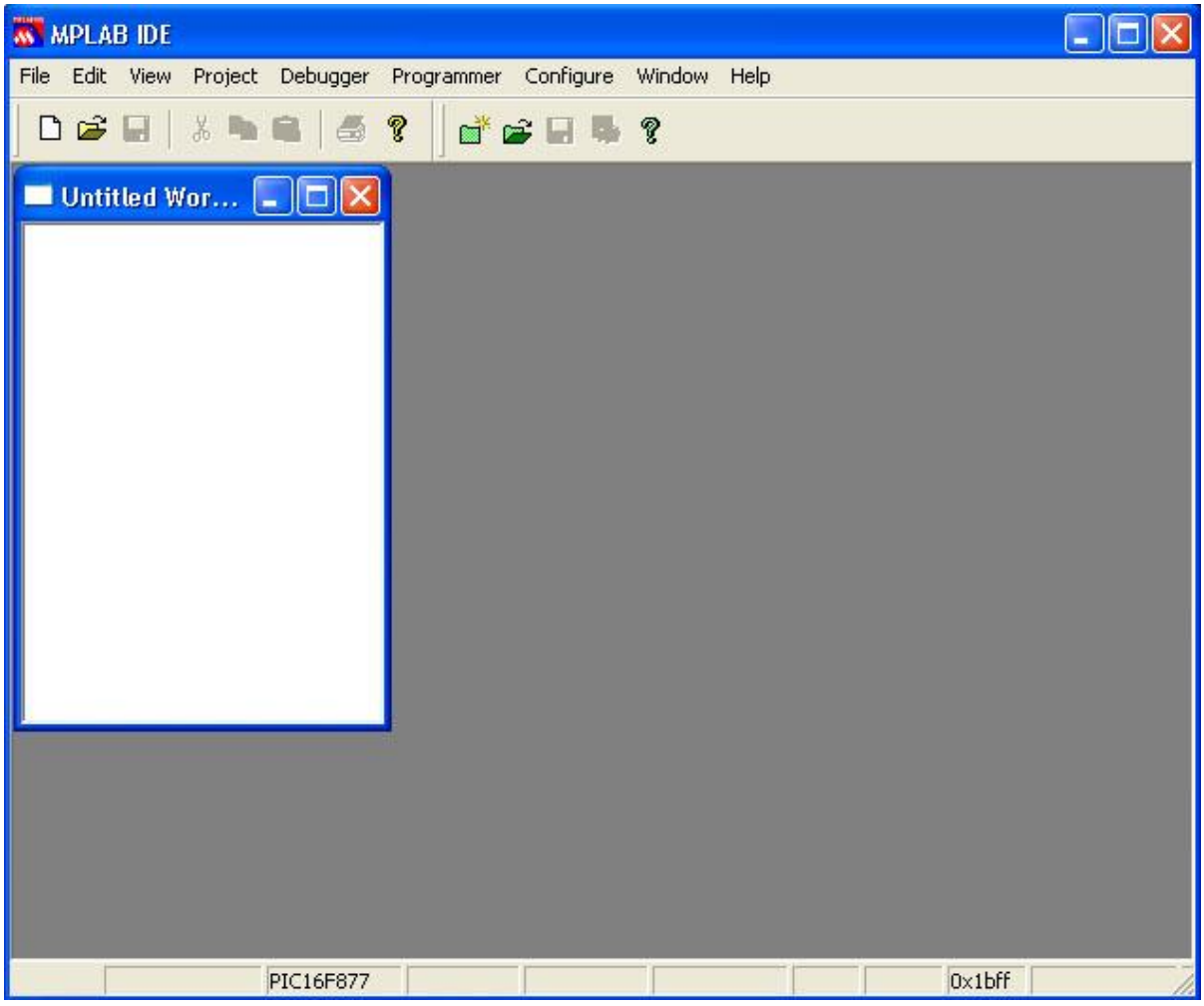
BÖLÜM 7. MPLAB PROGRAMININ KULLANIMI

7.1. 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.

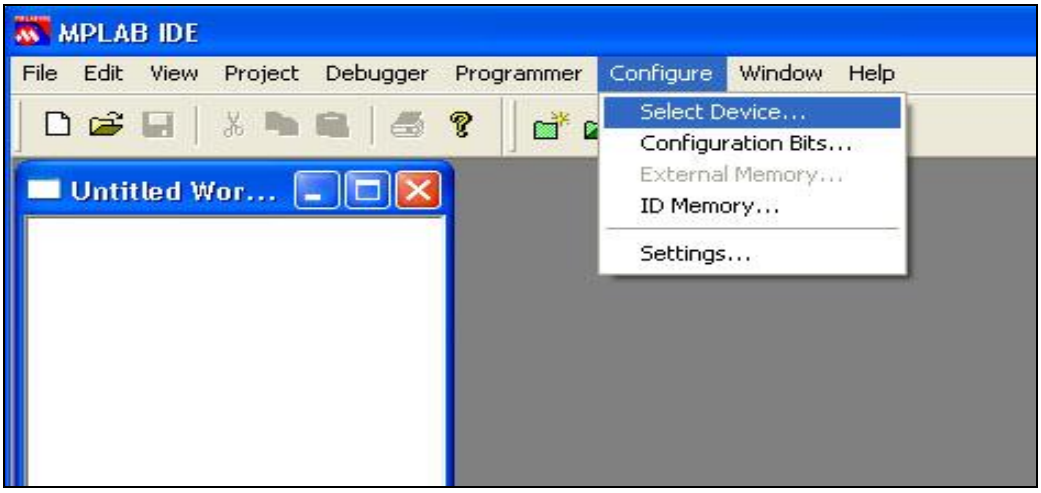
MPLAB programını *www.microchip.com* adresinde ücretsiz olarak indirebilirsiniz.

MPLAB programını bilgisayarımıza kurduktan karşımıza aşağıdaki pencere açılacaktır.



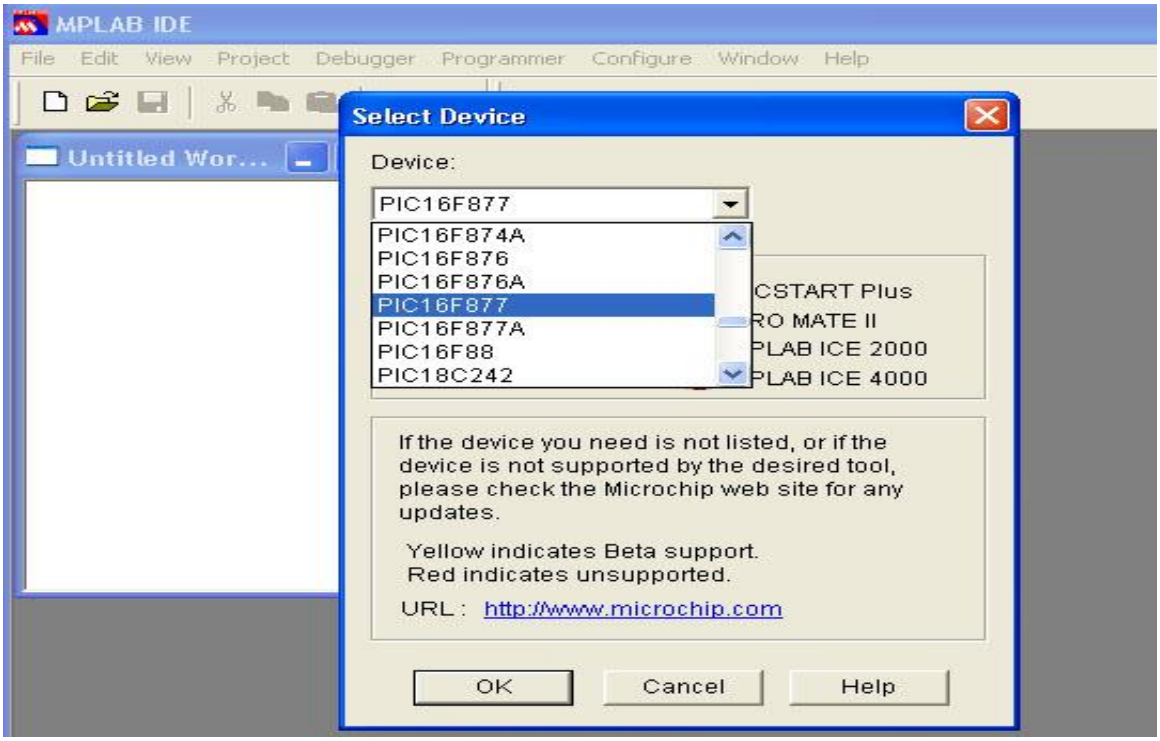
Şekil 7.1. MPLAB programı arayüzü

Öncelikle bizim burada kullandığımız PIC çeşidini seçip ayarlamamız gerekir. Bunu için önce Configure- Select Device seçeneğini seçmeliyiz.



Şekil 7.2 Aygıt seçimi

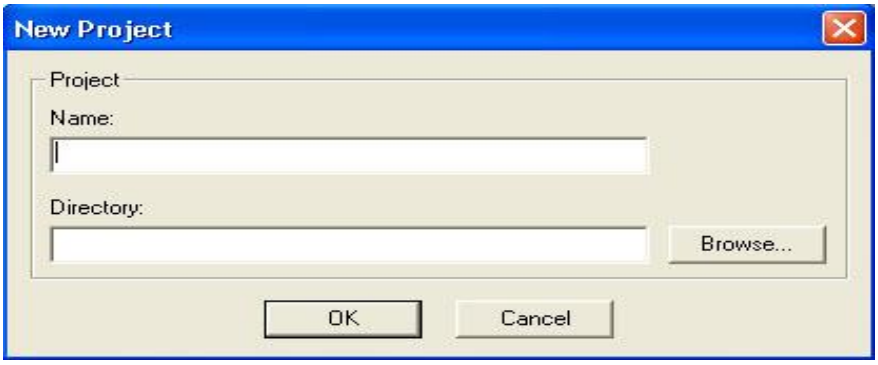
Bu seçimi yaptıktan sonra karşımıza aşağıdaki pencere karşımıza çıkacaktır. Buradan da PIc16F877 seçmemiz gerekir.



Şekil 7.3 PIC16F877 seçimi

7.2. MPLAB Programında Proje Oluşturma ve Derleme

MPLAB programı kullanılarak proje yapılmak isteniyorsa öncelikle yeni bir Proje dosya adı oluşturmak gerekir.

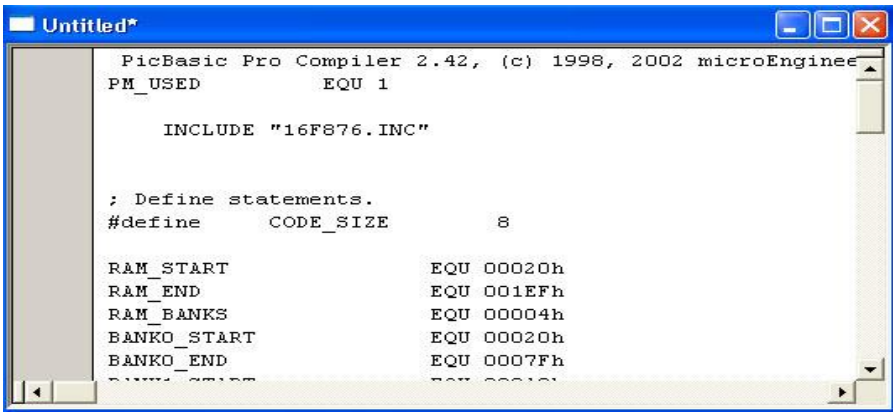


Şekil 7.4 Yeni proje oluşturma

Dosya adı oluşturulduktan sonra projenin kaydedileceği yer seçilir. Bu işlem yapıldıktan sonra projenin oluşturulması için belirtilen dizinde "mcp" ve "mcw" uzantılı iki dosya oluşturulur. Bu dosyalar proje(mcp) ve çalışma alanına (mcw) ilişkin bilgileri üzerinde tutan dosyalardır.

Şekil 7.5. Proje oluşturduktan sonraki çalışma alan penceresi

Daha sonra File menüsünden New seçeneğini seçerek yeni açılan Untitled penceresine program kodları yazılır.



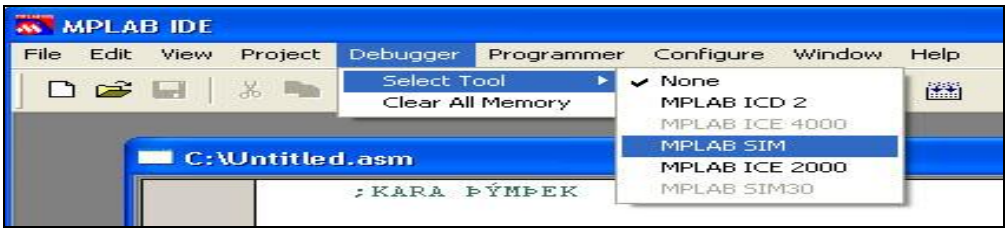
Şekil 7.6. MPLAB metin editörü

Program yazıldıktan sonra derlenmesi ve bir *HEX* dosyasının oluşturulması gerekir. Bunun için File menüsünden Save seçeneği seçilir. Açılan pencerede, dosya uzantısının ASM olması gerekir. Derleme aşmasında MPASM programının ayarlarını yapmak için Project menüsünden Build Option seçilir.

Şekil 7.7. Kısayol tuşları

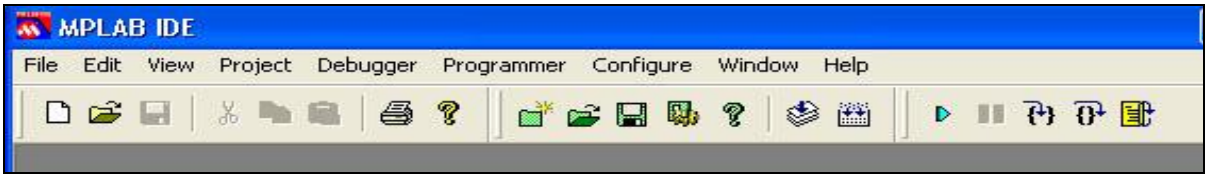
MPLAB ile Simülasyon

MPLAB ile simülasyon yapmaya başlamadan önce bazı ayarlamaların yapılması gerekir. Bunun için öncelikle Debugger menüsüden Select Tool seçilerek açılan yeni pencerede MPLAB SIM seçeneği seçilir.



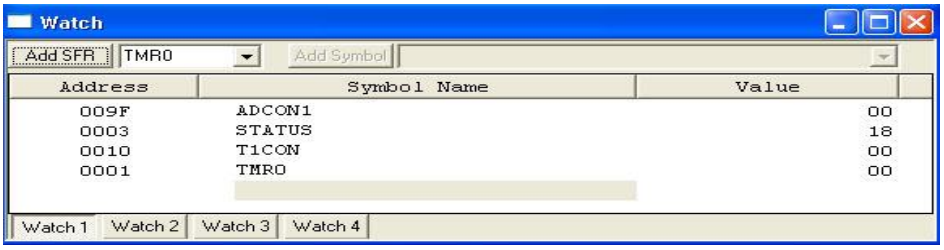
Şekil 7.8. Simülasyon için ayarların yapılması

Bu işlem yapıldığında benzetimle ilgili kısayol tuşları eklenir.



Şekil 7.9. Benzetim işlemi için kullanılabilecek kısayol tuşları

Programı bu kısayollar üzerinden kontrol edebiliriz. Program kodları çalışırken saklayıcıların, program ve veri belleğinin durumu gözlenebilir. Bunun için View menüsünden Watch özelliği kullanılır.

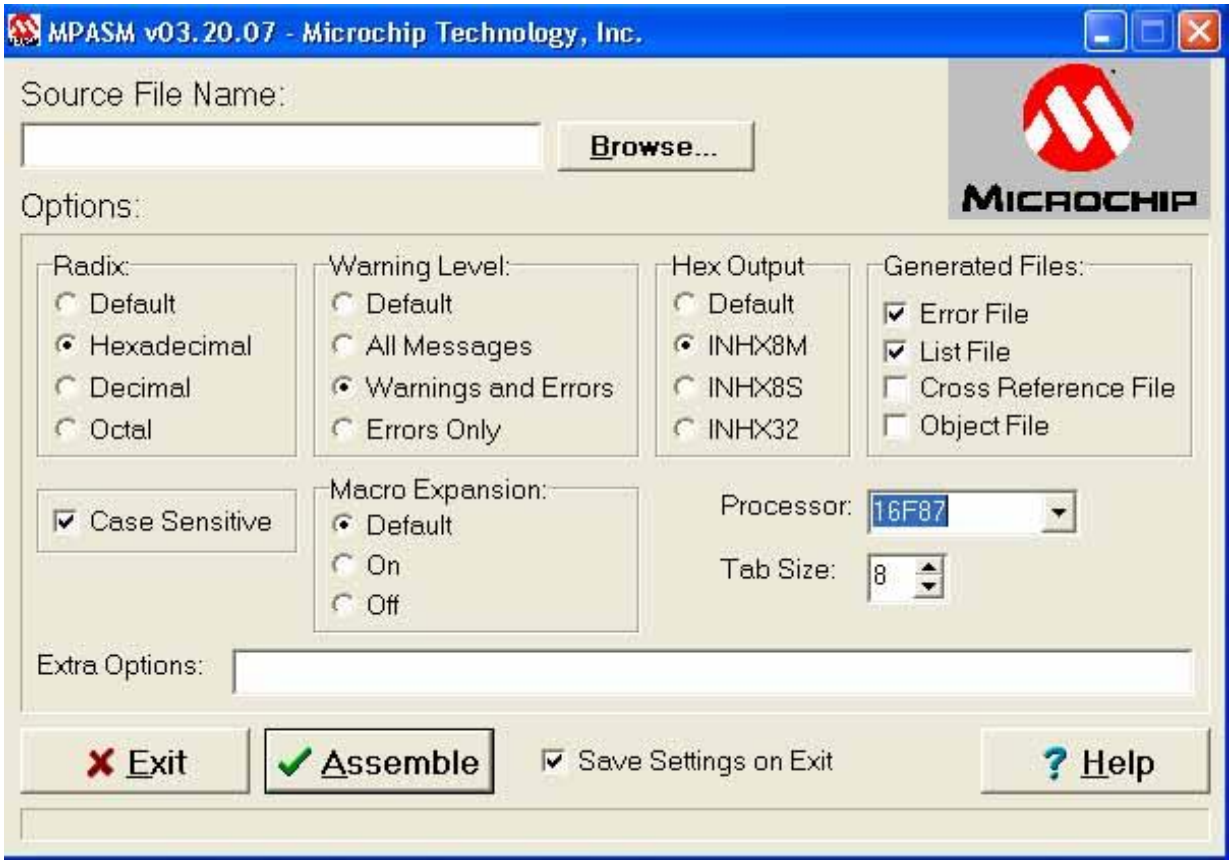


Şekil 7.10. Watch Penceresi

Program çalışırken değeri gözlemlenmek istenen saklayıcı Add SFR butonunu kullanarak izleme penceresine eklenir. Value kısmında ise seçilen saklayıcıların durumları yer almaktadır.

MPASM

Yazdığımız program kodlarınız aynı zamanda MPASM programını kullanarak da derleyebiliriz. Şekil 7.11 ‘de MPASM programının arayüzü görülmektedir. Bu program PIC assembly dilinde yazılan program kodları PIC ‘in çalıştırabileceği HEX kodlarına dönüştürmeyi sağlar.



Şekil 7.11. MPASM arayüzü

Bunun için öncelikle .asm uzantılı kaynak dosya gerekmektedir. PIC program kodları yazılırken istenilirse not defterinden yazılıp uzantısı asm olarak kaydedildikten sonra buradan derlenmesi yapıla bilinir. Browse butonu kullanılarak derlenmek istenen asm dosyası seçilir. Options kısmından gerekli ayarları yaptıktan sonra "Assemble" butonuna basılarak ilgili programın Hex kodları oluşturulur.

Radix bölümünde programın yazılımı sırasında tipi belirtilmeyen sayılar varsa burada seçilen tipe göre işleme koyulur. Örneğin hexadecimal seçeneği seçilmişse tipi belirtilmeyen sayılar hexadecimal olarak kabul edilir.

Warning level, program derlenirken oluşabilecek hata ve uyarıları kullanıcıya bildirmek için kullanılır. "Warning and Errors" seçeneği seçildiğinde sadece derleme sırasında karşılaşılan uyarı ve hata durumlarında kullanıcı bilgilendirilir.

Hex Output seçeneğinde Hex dosyasının tipini belirtmek için kullanılır.

Case Sensetiev seçeneği de programda kullanılan değişkenlerin küçük/büyük harfe duyarlı olup olmaması sağlanır.

Macro Expension, eğer programınızda macro kullandıysanız buradaki ayarların yapılması gerekir.

Processor kısmında ise kullandığınız PIC mikrodenetleyicisi seçebilirsiniz. Buradan PIC 16F877 seçilmelidir.

Tab Size , sütunlar arası mesafeyi ayarlamak için kullanılır.

Bütün bu seçimler yapıldıktan sonra "Assemble" butonuna tıkladığınız zaman program derlenmeye başlanacaktır. Program derlendikten sonra programımızı PIC'e yükleyebiliriz.

7.5. Bölüm Kaynakları

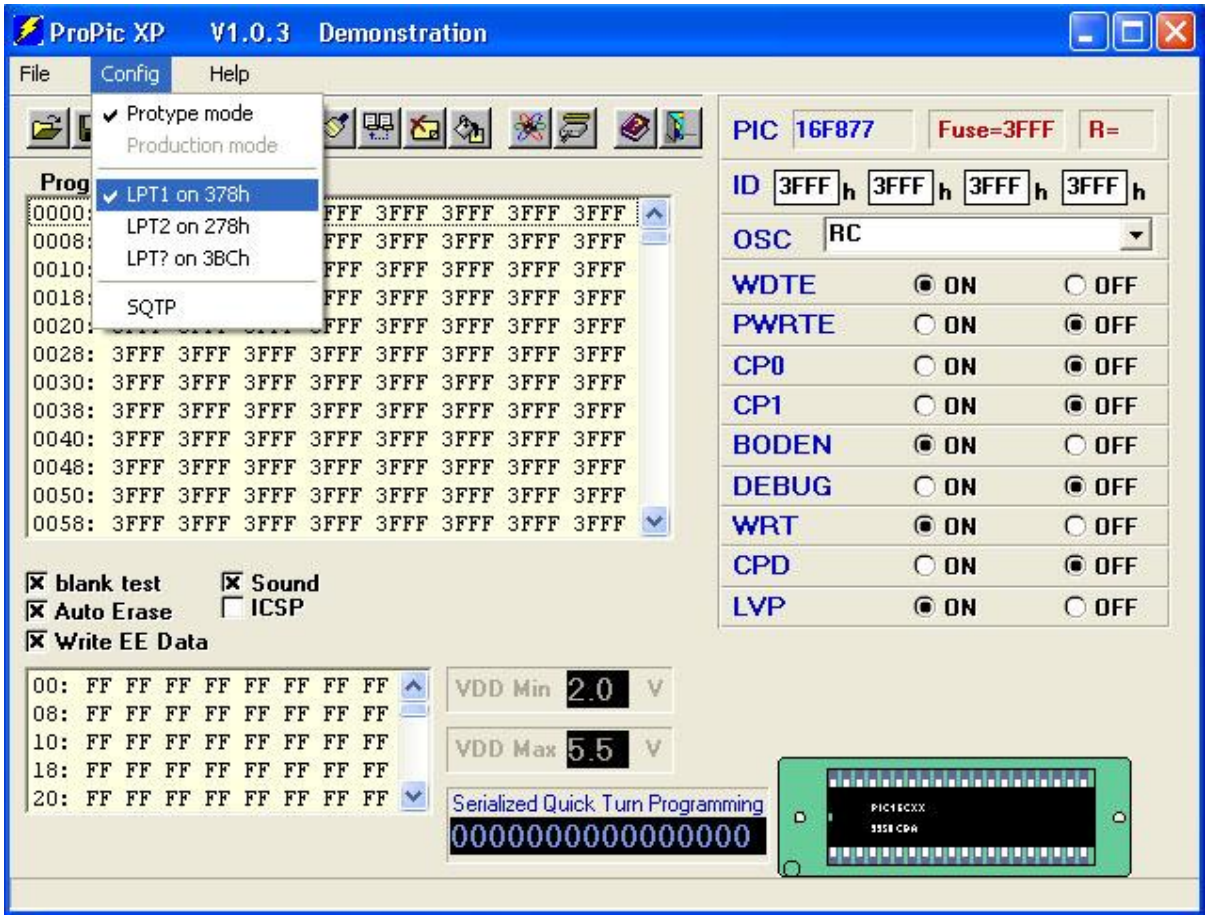
1. *O. Altınbaşak, 2001. "Mikrodenetleyiciler ve PIC Programlama", Atlaş Yayıncılık, İstanbul.*
2. *O. Urhan, M.Kemal Güllü, 2004. "Her Yönüyle PIC16F628", Birsen Yayınevi, İstanbul.*
3. *Y. Bodur, 2001. "Adım Adım PICmicro Programlama",İnfogate.*
4. *Ç. Akpolat, 2005."PIC Programlama", Pusula Yayıncılık*

BÖLÜM 8. PIC PROGRAMLAYICI

8.1. PROPIC Programı

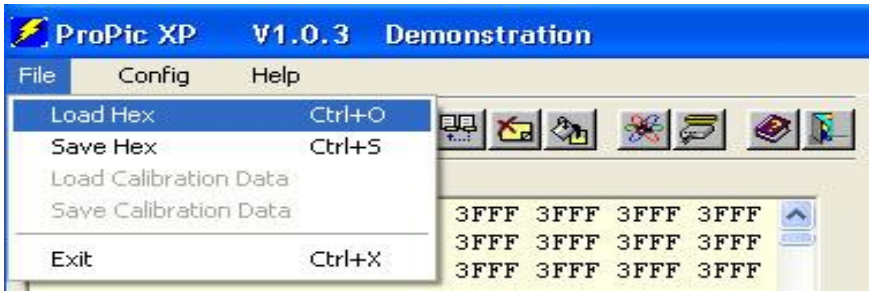
PIC programlayıcının kullanılabilmesi için, Win98 ve üstü bir işletim sistemi, paralel port gerekmektedir. Programımızı derlediğimizde *HEX* uzantılı bir dosya oluşturulacaktır. PIC Programlayıcı ile işlemciye yazılacak dosya da bu dosyadır. Programı kurup, programlayıcıyı bilgisayara bağladıktan sonra hazırlamış olduğunuz *HEX* dosyasını PIC yüklemeniz gerekir.

MPLAB'i kullanarak uygulamanızı geliştirin ve derleyin. Derleme işlemi *.HEX* uzantılı makine kodu içeren bir dosya oluşturacaktır. Programlayıcı devreyi bilgisayarın LPT (Printer) portuna, programlayıcı devreyi de güç kaynağına bağlamamız gerekecektir.



Şekil 8.1. PROPIC Yazılımı

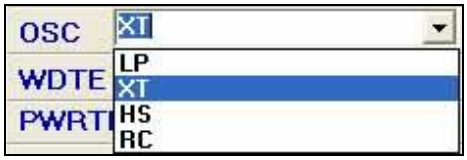
PROPIC bilgisayarın çeşitli portlarını kullanabilmektedir. Bizim kullandığımız programlayıcı devrenin doğru çalışabilmesi için doğru portu seçmemiz gerekecektir. Bunun için Config menüsünden LPT1 on 378h seçilir. Bu seçim işlemini yaptıktan sonra yan tarafta yer alan PIC isimleri içerisinden kullandığınız PIC seçilir.



Şekil 8.2. HEX dosyasının yüklenmesi

Seçim işlemleri yapıldıktan sonra yazılan hex dosyasını PIC'e yüklemek için File menüsünden Load Hex seçeneği seçilir. Yazmış olduğunuz dosyayı seçtikten sonra ok dediğinizde dosyanız PROPIC içerisine dahil edilir.

Kullandığımız PIC 16f877 de kristal osilatör kullandığımız için XT seçeneği seçilir.



Şekil 8.3. Osilatör seçimi

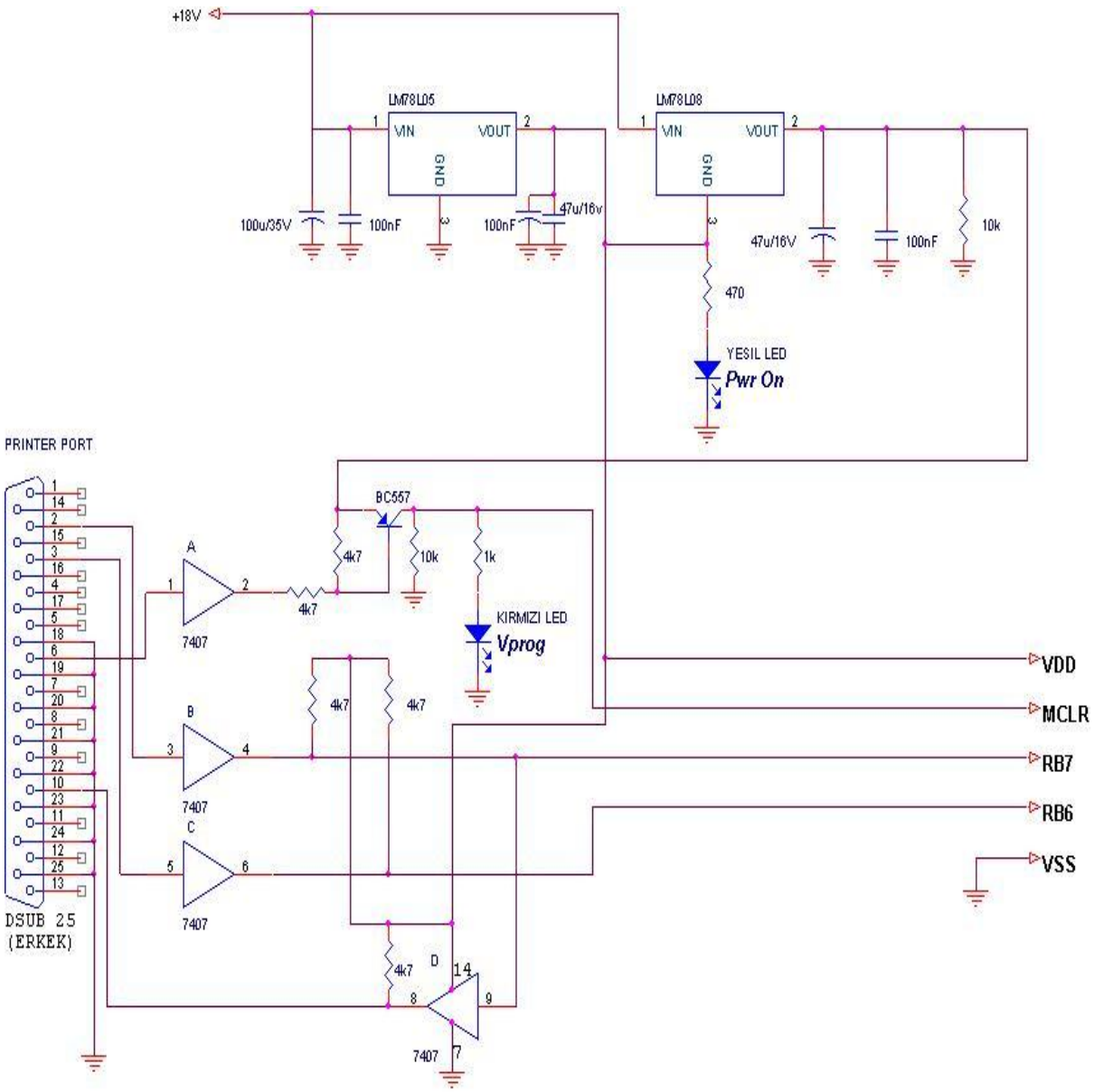
 simgesi ile PIC içerisindeki bilgiler okunur.

 simgesi ile PIC içerisine derlemiş olduğumuz HEX dosyası yüklenir.

 PIC içerisindeki bilgilerin silinmesini sağlar.

PIC içerisine HEX uzantılı dosyamızı yükledikten sonra PIC' hazırladığımız devrede kullanabiliriz.

8.2. PIC Programlayıcı Devresi



Şekil 8.4. Programlayıcı

Yukarıda görülen devre şeklinden faydalanarakta PIC'i programlayabiliriz.

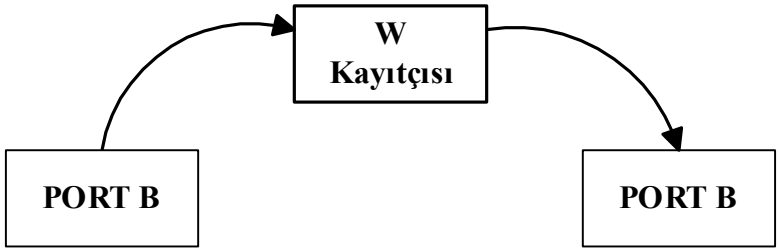
8.3. Bölüm Kaynakları

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. *Y. Bodur, 2001. "Adım Adım PICmicro Programlama",İnfogate.*
4. *Ç. Akpolat, 2005."PIC Programlama", Pusula Yayıncılık*

BÖLÜM 9. PIC PROGRAMLAMA

9.1. 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.
```

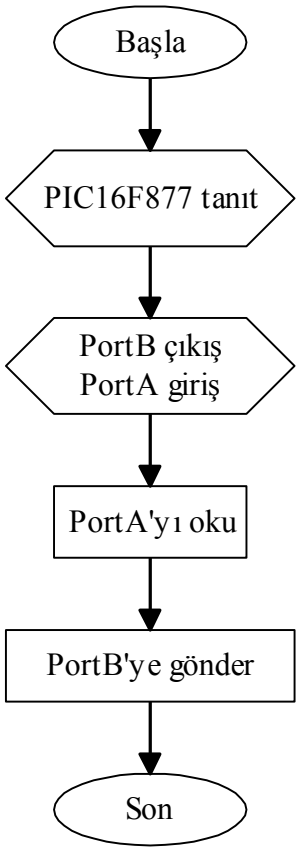
Burada W kayıtçısına gönderilen h '0F' verisini binary karşılığı b'00001111' dir. Bu veriye göre PortB'nin ilk 4 biti bağlı olan ledler yanar.

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 9.1:

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

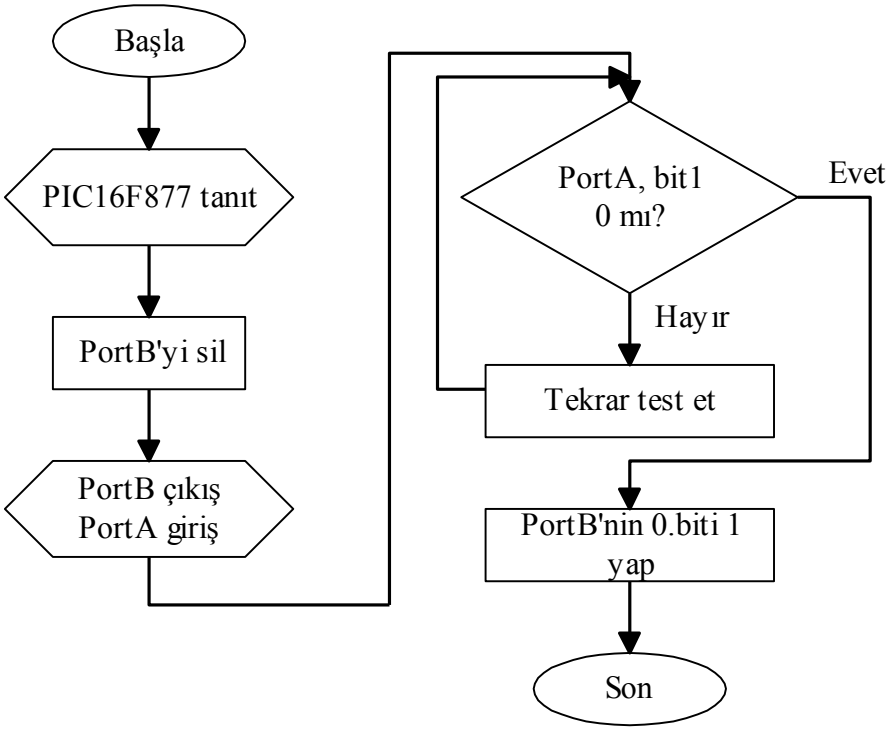



```
            LIST        P=16F877
PORTA       EQU         h ‘05’
PORTB       EQU         h ‘06’
STATUS      EQU         h ‘03’
TRISA       EQU         h ‘85’
TRISB       EQU         h ‘86’
            CLRF        PORTB       ; PortB’ye bağlı ledleri söndür
            BSF         STATUS,5    ; BANK1’e 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 9.2: PortB'nin 0. bitine bağlı Led'i, A portunun 1. bitindeki butona basılınca yakan program.



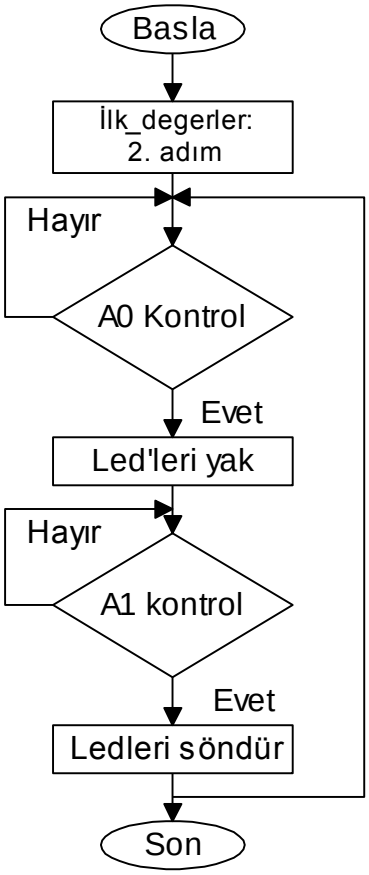

```
        LIST    P=16F877
PORTA   EQU     h '05'
PORTB   EQU     h '06'
STATUS  EQU     h '03'
TRISA   EQU     h '85'
TRISB   EQU     h '86'
        CLRF    PORTB       ; PortB'ye bağlı ledleri söndür
        BSF     STATUS,5    ; BANK1'e geç
        CLRF    TRISB       ; PORTB'nin uçlarını çıkış 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 9.3: 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
```

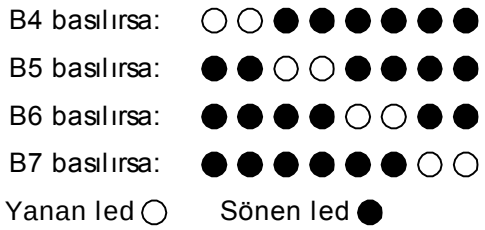
Örnek 9.4: PORTA ‘nın 0. bitine bağlı butona basıldığında B portuna bağlı ledlerin yanmasını, 1. bitine bağlı butona basıldığında B portundaki ledleri söndüren program.

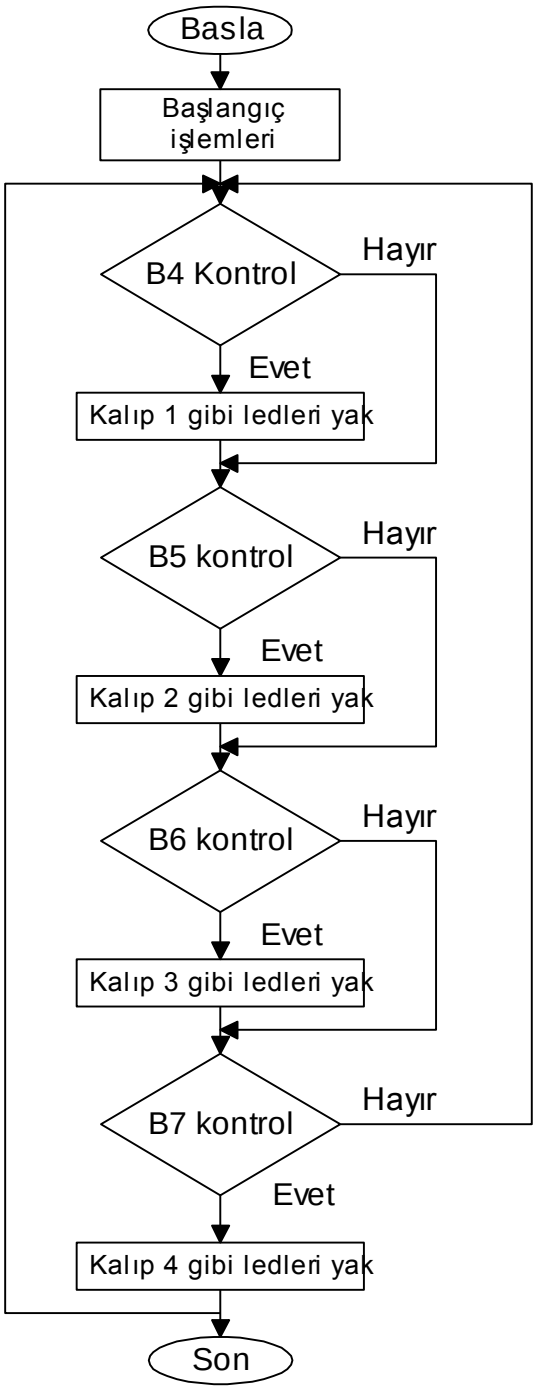



```
            LIST    p=16F877
            INCLUDE “p16F877”
            Org     0x00
            Goto    Basla
Basla
            CLRF    PORTA           ; PORTA yazmacını temizle
            CLRF    PORTB           ; PORTB yazmacını temizle
            BSF     STATUS,5        ; Bank1 e geç
            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 9.5: PIC16F877 PortB ‘nin 4-7. bitlerine bağlı butonları kullanarak; PortD’ye bağlı sekiz ledi aşağıdaki yakan programı yazın.





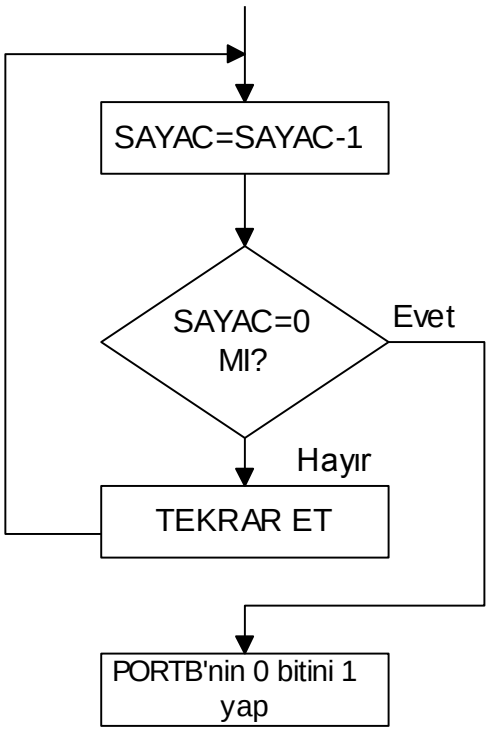
```
            LIST        p=16F877
            INCLUDE     “p16F877”
KALIP1      EQU         H ‘C0’          ; b ‘11000000’
KALIP2      EQU         H ‘30’          ; b ‘00110000’
KALIP3      EQU         H ‘0C’          ; b ‘00001100’
KALIP4      EQU         H ‘03’          ; b ‘00000011’
#DEFINE     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
```

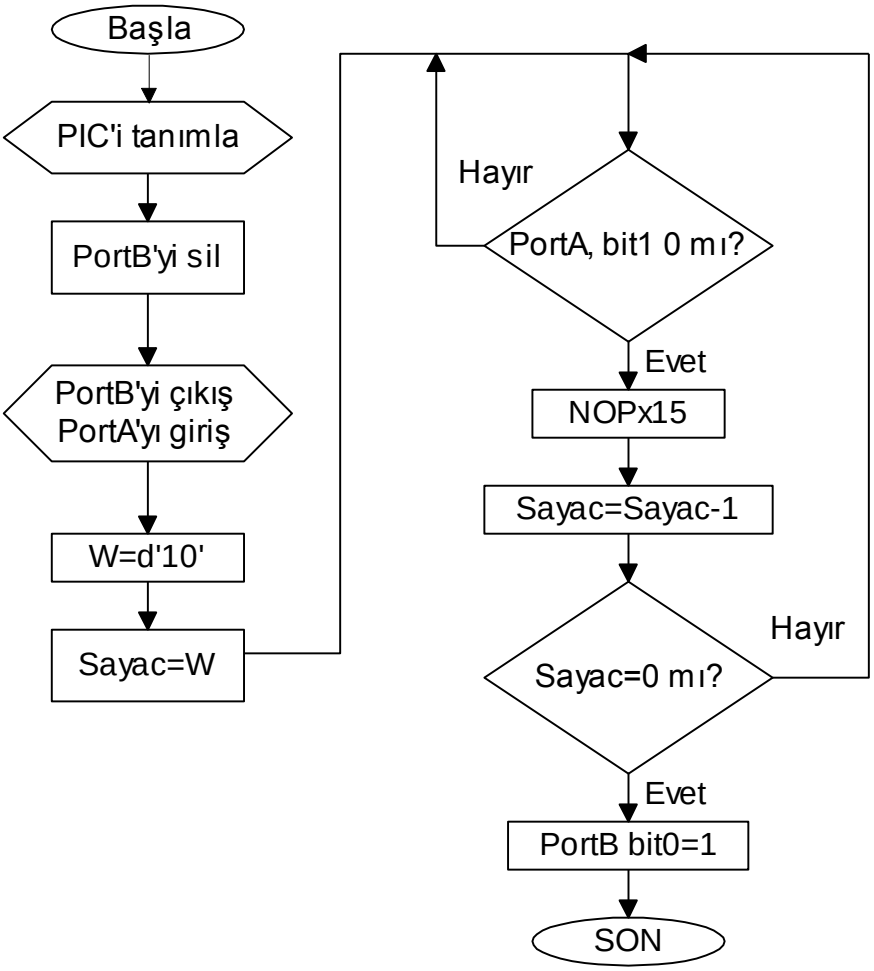
9.2. Döngü Düzenlemek

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
```

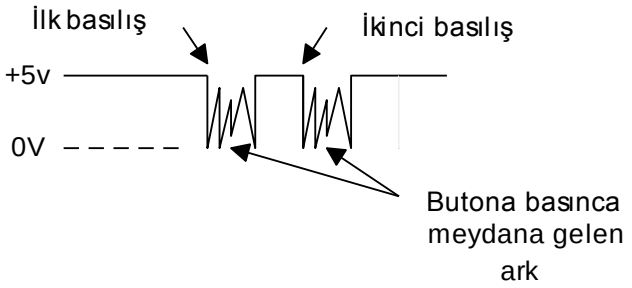


Örnek 9.6: A portunun 1. bitine bağlı butona 10 defa basıldıktan sonra B portunun 0. bitine bağlı olan ledleri yakan program ve akış şeması



```
        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
```

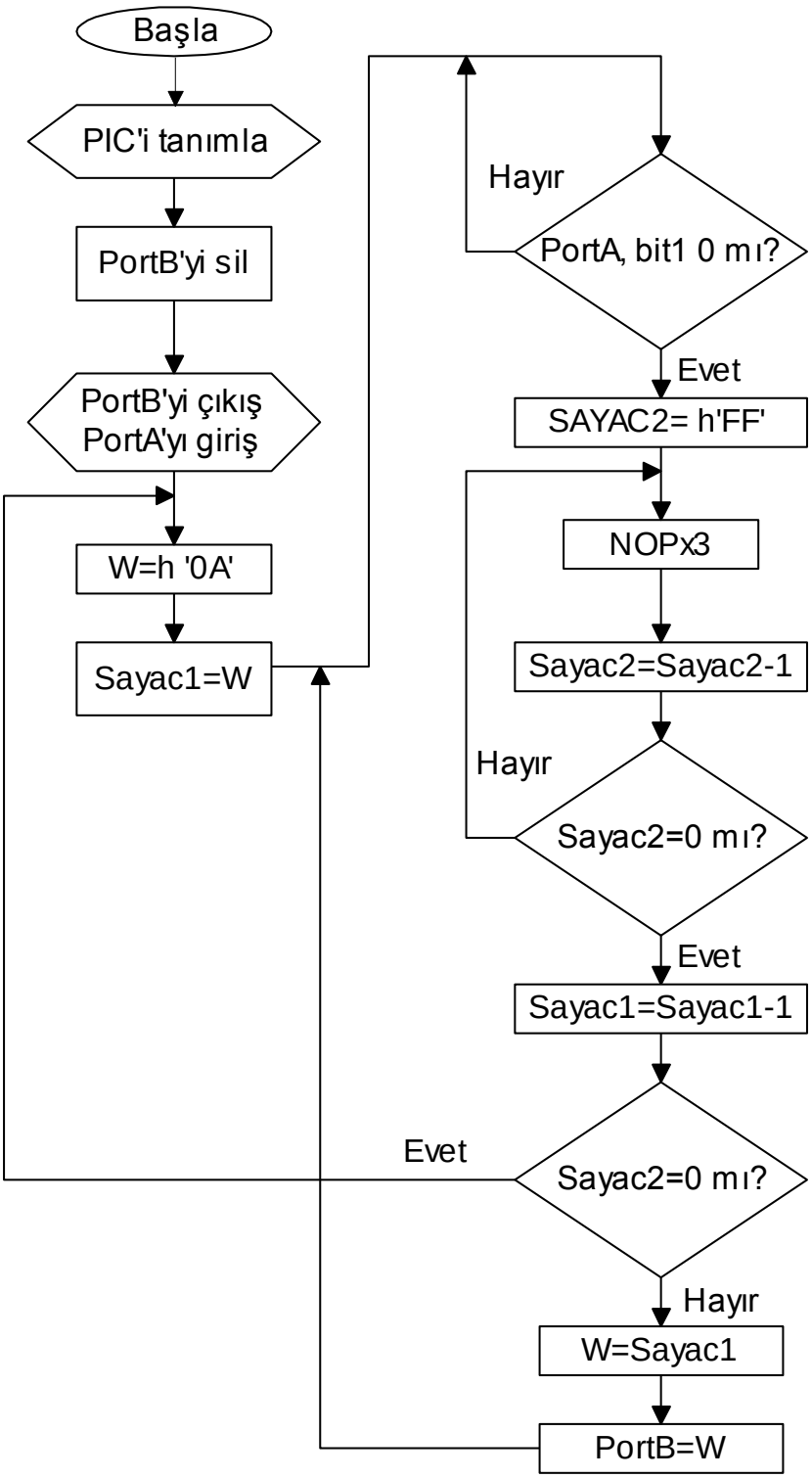
Butona basma sayısı 10'a ulaşmadan PortB'nin 0.bitindeki LED'in yandığı görülmektedir. Burada pull-up olayı gerçekleşmiştir. Yani butona basılmadığında +5 voltta basıldığında ise 0 V olmaktadır. Butona basma ve çekme esnasında bir ark oluşur.



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

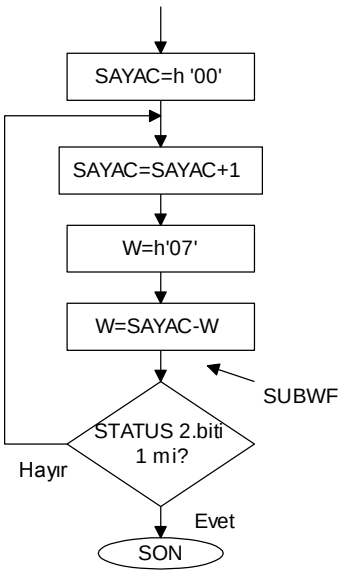
arka arkaya defalarca basılmış gibi işlem göstererek DECFSZ SAYAC,F komutuyla SAYAC kayıtçısının değeri her defasında 1 olacaktır. Butonun meydana getireceği ilk 0 seviyesine düşüşten sonraki 0 durumlarını eleyip tekrar 5V durumuna kadar belirli bir gecikme oluşturmak gerekecektir. Bu gecikmeyi NOP komutu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

Örnek 9.7: 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
```

9.3. 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.

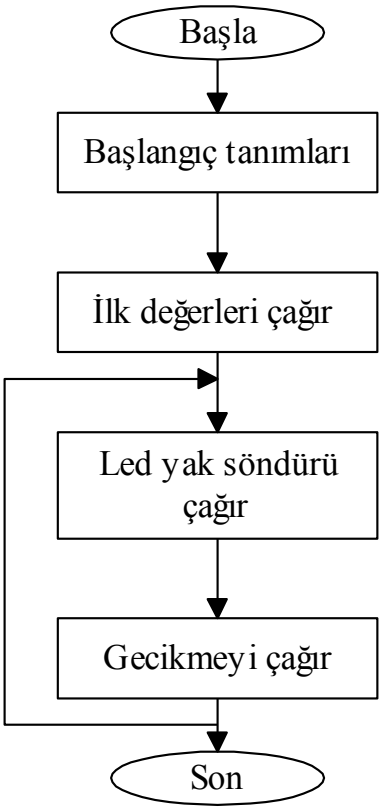
Örneğin NOP komutu ile 0,1 milisaniyelik bir gecikme yaratmak için ne kadar NOP komutu gerekir. (Kristal Osilatör → 20 Mhz)

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 9.8: 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.

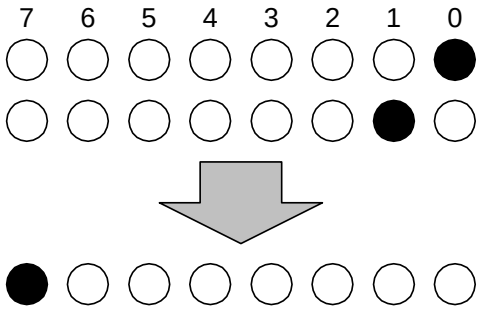
Örnek 9.9: PortA'nın 1. ucuna bağlı butona 10 defa basıldıktan sonra PortB deki tüm ledleri yakan program.

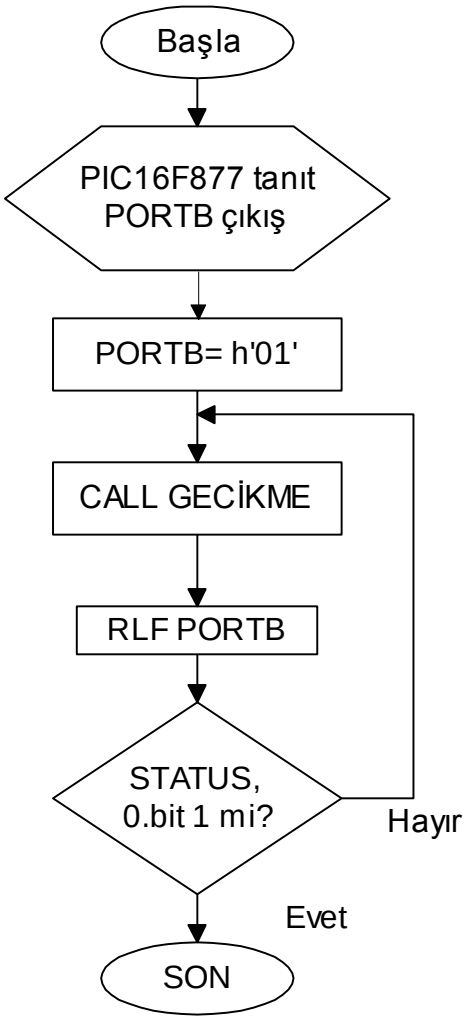
```
            LIST    P16F877
            INCLUDE "P16F877.INC"
SAYAC1      EQU     h '20'
SAYAC2      EQU     h '21'
MEM         EQU     h '22'
            CLRF    PORTB       ; PortB'yi sil
            BSF     STATUS,5    ; Bank1'e geç
            CLRF    TRISB       ; PortB'nin uçları çıkış
            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
```

9.4. Bit Kaydırma

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

Örnek 9.10: PortB'ye bağlı 8 led üzerindeki bir ledin yanışını LED0'dan LED7'ye doğru kaydıran program ve akış şeması.



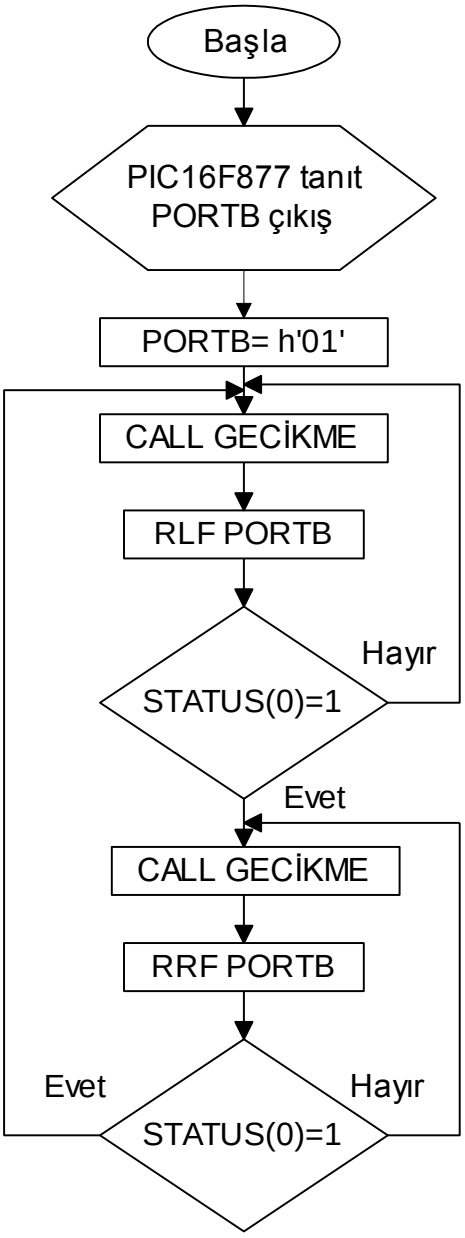



```
            LIST        P16F877
            INCLUDE     "P16F877.INC"
SAYAC1      EQU         h '20'          ;SAYAC1 'e adres atandı
SAYAC2      EQU         h '21'          ;SAYAC2 'e adres atandı
            CLRF        PORTB           ; PortB ye bağlı ledleri söndür
            BCF         STATUS,0        ; 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
```

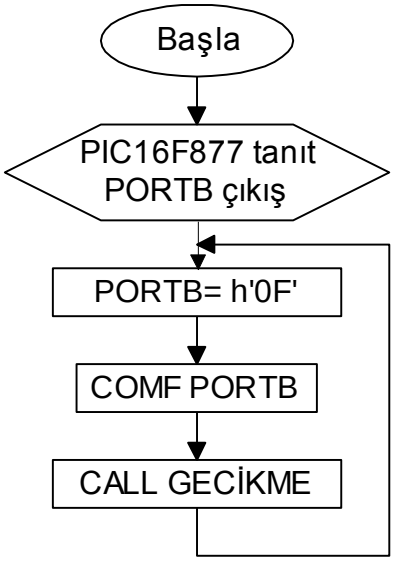
Örnek 9.11: PortB’ye bağlı olan 8 LED üzerinde bir LED’in yanışını sağa-sola kaydıran ve bu işlemi sürekli tekrarlayan program ve akış şeması.**(Karaşimşek devresi)**



```
            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 9.12: 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
```

9.5. 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 9.13: 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
```

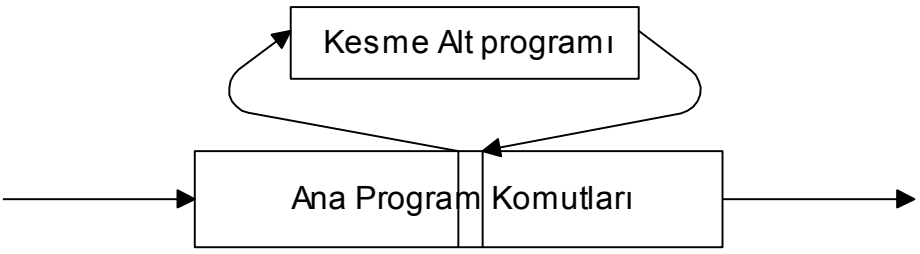
Örnek 9.14: PORTB'nin uçlarına bağlı 7 segment display'de 0~F arasında saydıran program.

```
        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
```

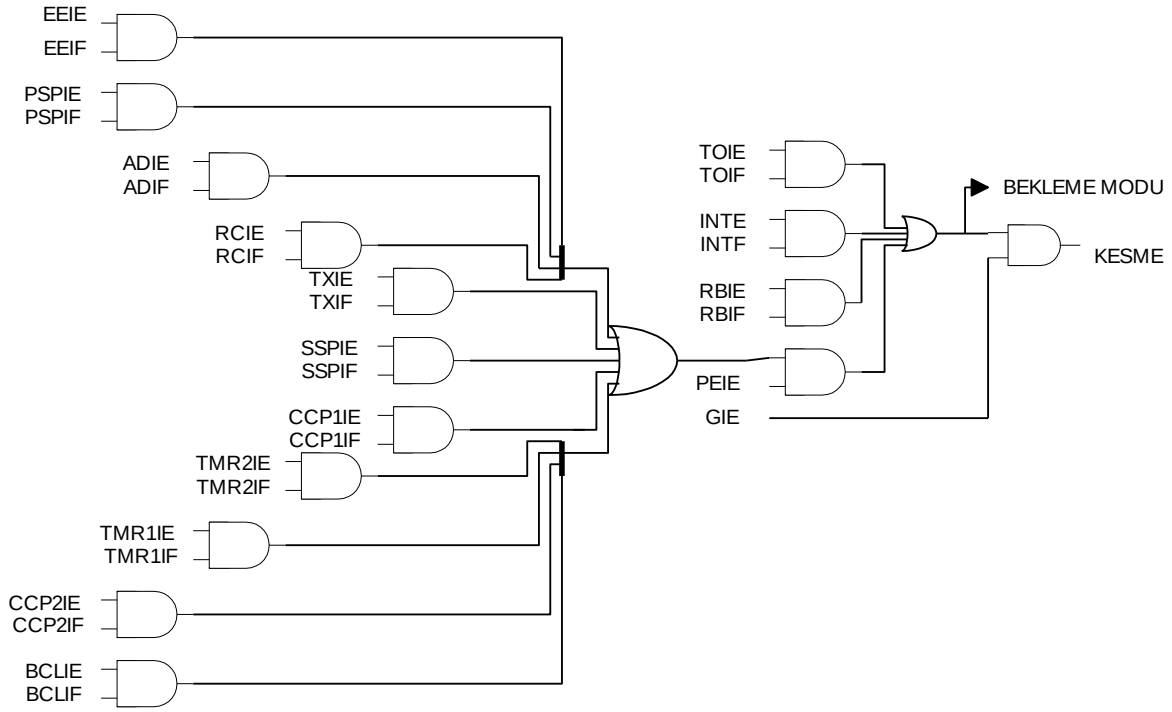
9.6. Kesmeler

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.

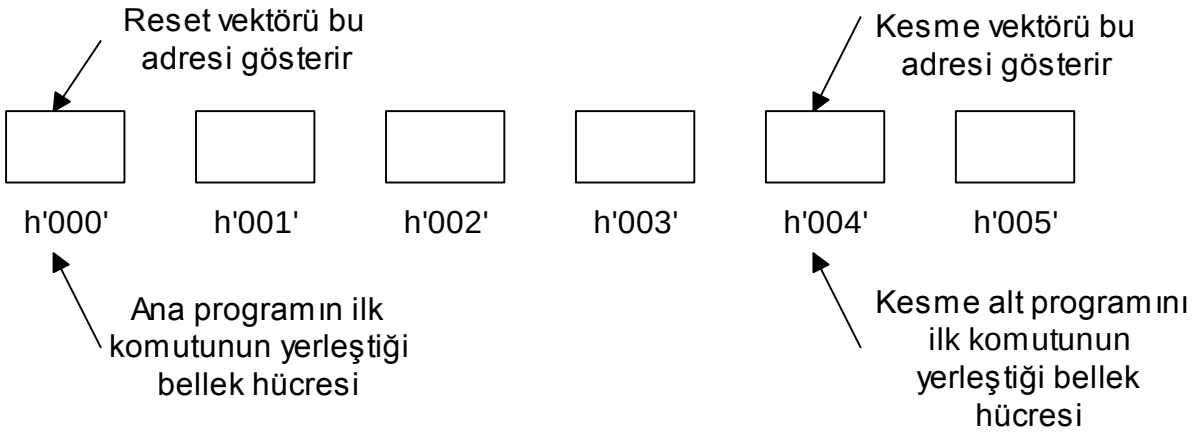


Device	TOIF	INTF	RBIF	PSPIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	EEIF	BCLIF	CCP2IF
PIC16F876-873	+	+	+	-	+	+	+	+	+	+	+	+	+	+
PIC16F877-874	+	+	+	+	+	+	+	+	+	+	+	+	+	+

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 9.2. 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
```

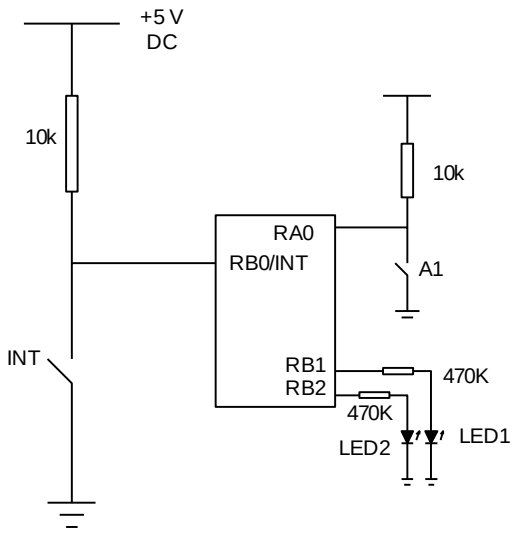
Bir kesme oluştuğu zaman kesme alt programı çalışmadan önce gecikme meydana gelir. Bu gecikme süresi 3 yada 4 komut saykılı süresindedir. Zamanlamanın çok önemli olduğu uygulamalarda bu zaman gecikmesi dikkate alınmalıdır.

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.

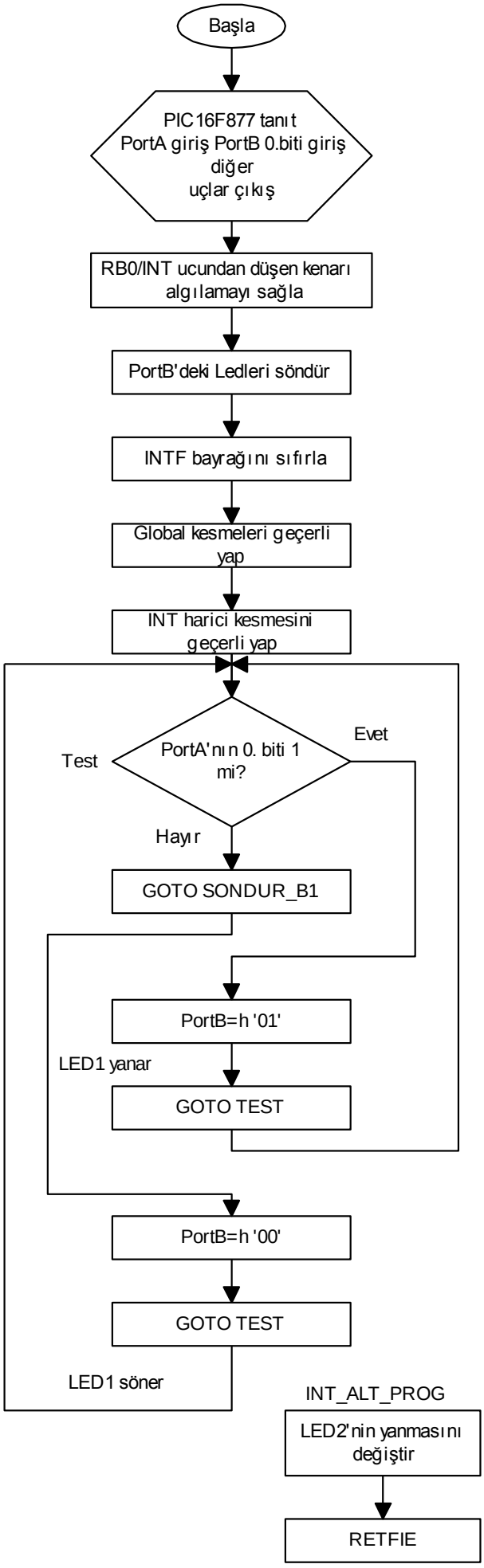
Kesme kullanılırken; PCL, Status ve W yazmaçlarının içindeki değerleri koruma işi de programlayıcıya bırakılmıştır. Bu yazmaçları korumak için PIC veri sayfalarında aşağıdaki komutları kullanmak yeterlidir.

```
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 9.15 : 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.

9.7. Zamanlayıcılar

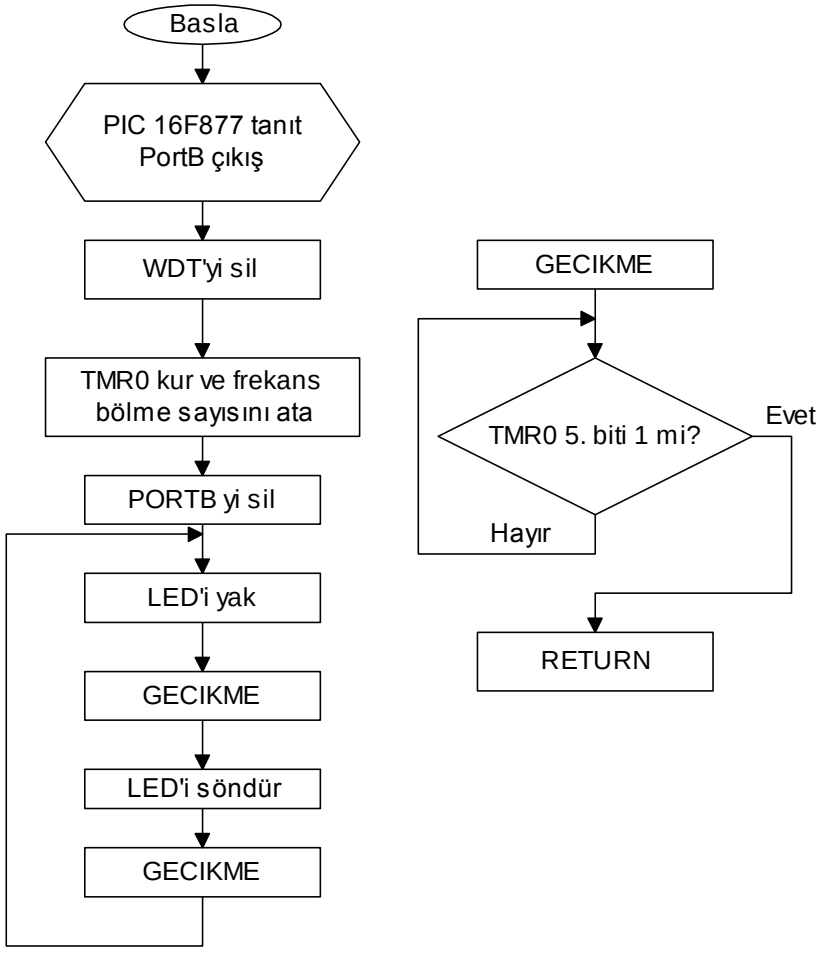
PIC16F877 ailesinde Timer0, Timer1, Timer2 ve WDT adları verilen 4 tip zamanlayıcı bulunmaktadır.

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;

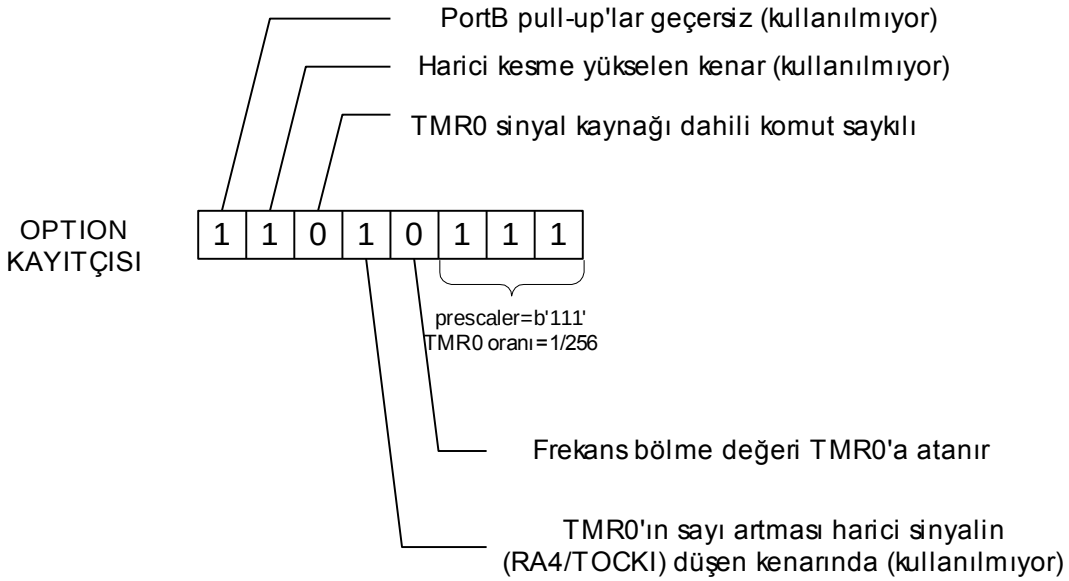
- 8 bitliktir.
- Herhangi bir anda sıfırlanabilir.
- Üzerine yazılabilir veya okunabilir.
- Programlanabilir frekans önbölücü değeri kullanılabilir.
- İçindeki veya dışındaki devrede bulunan osilatör saatleri kullanabilir.
- Dış sinyallerle düşen veya yükselen kenar tetiklenmesini yapabilir.
- Sayacı 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.

9.16. 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;

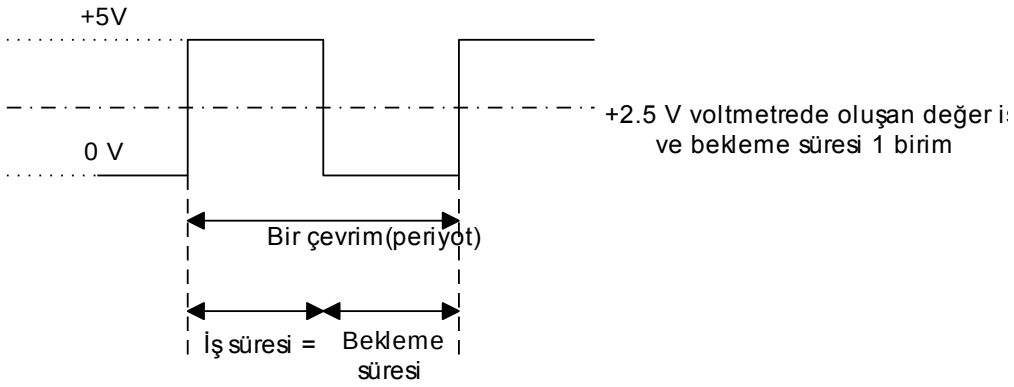
256µS x 32 =8192µS →8.2 msn (4 MHz kristal osilatör kullanıldığı düşünülürse)

9.8. A/D VE D/A Dönüştürme

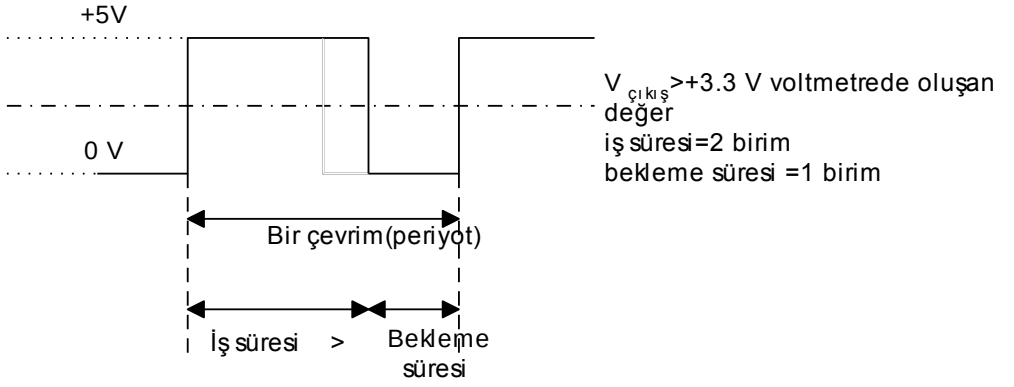
PIC ile yaptığımız uygulamalarda şimdiye kadar port çıkışlarında sayısal sinyaller elde ettik. Ancak bazı uygulamalarda da bize analog sinyaller gerekmektedir. Analog sinyalleri oluşturmak için dönüştürücüler kullanılır. Bu bölümde de A/D ve D/A dönüştürme işlemlerini göreceğiz.

9.8.1. PWM Metodu ile D/A Dönüşümü

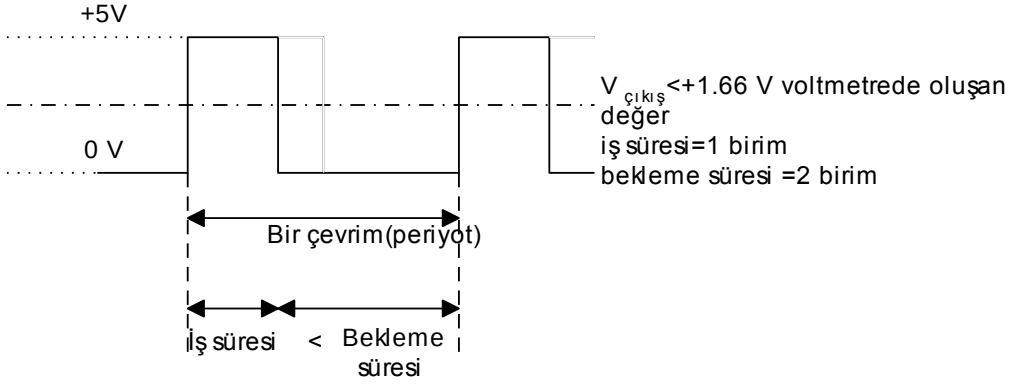
PWM yöntemi, sinyalin iş çevrim süresini değiştirerek sinyalin çıkışında bir kare dalga sinyal yaratma yöntemidir. İş çevrim süresi, sinyalin ledin gerçekten yandığı +5V olduğu aralıktır. Ledleri yakıp söndürürken bir kare dalga sinyali kullanılır. Ledin ucuna bir voltmetre bağladığımız takdirde gerilimi +2.5 V olarak ölçeriz. Halbuki bu sinyalin bir çevrim sürelik zamanın ilk yarısında gerilim +5V iken diğer yarısında 0V tur. Süre bakımından çok kısa olduğu için gözümüz bunu led yanıyor gibi algılar.



A



B



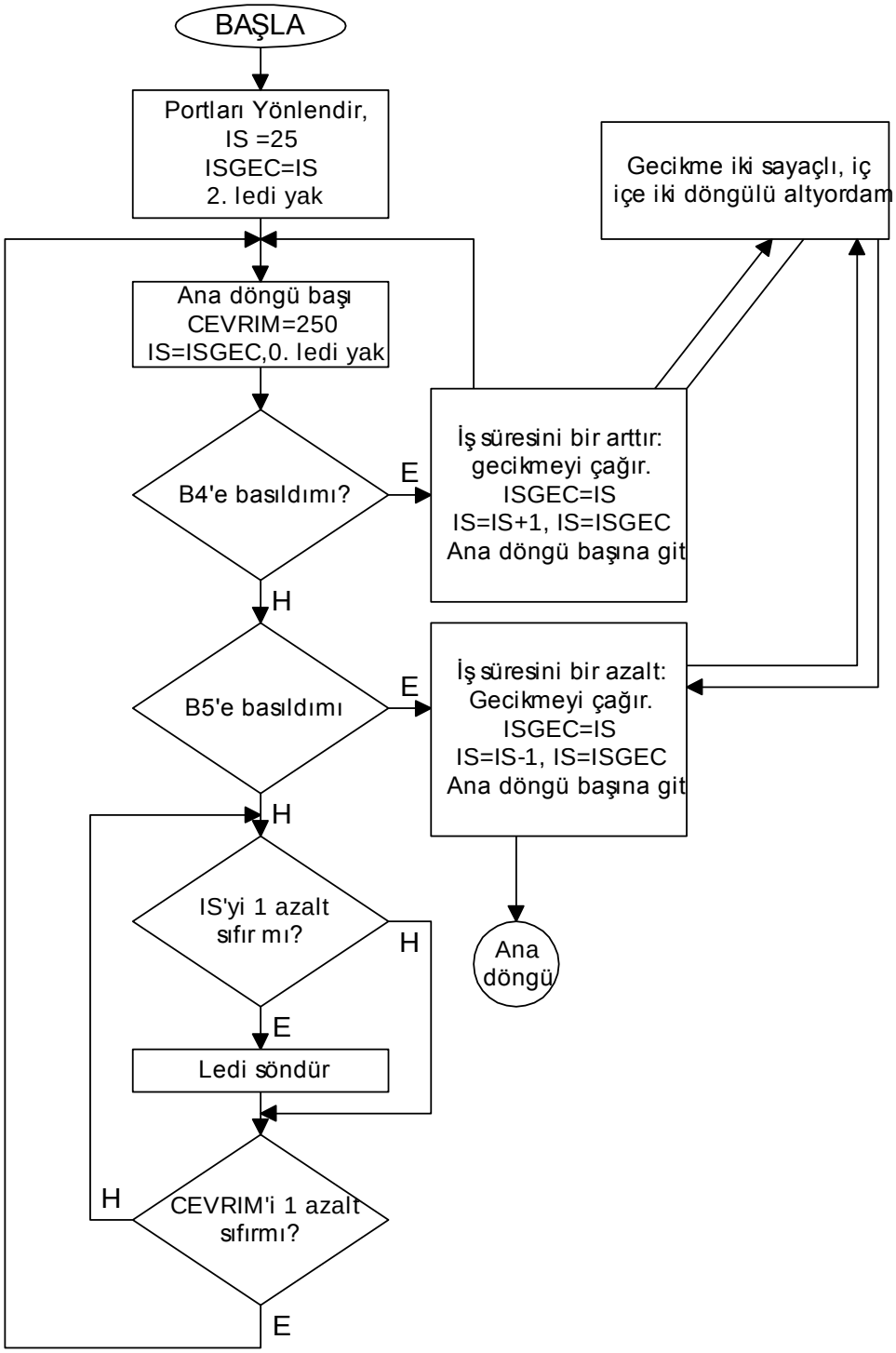
C

Yukarıdaki A,B ve C şekillerinde çıkış gerilimini hesaplarsak.

- A=2,5 V
- B şeması için Çıkış Gerilimi= ((5V)*2)/3=(10)/3=3,33V
- C şeması için Çıkış Gerilimi= ((5V)*1)/3=(5)/3=1,66V

Çıkış pinindeki gerilimi, gecikme ile değiştirerek bir ledin ışığının parlaklığı ayarlanabilir.

Örnek 9.17. PortD'nin 0. bitine bağlı ilk ledin parlaklığını; PortB'nin 4. bitine bağlı B4 butonuna basıldığında arttıran 5. bitine bağlı B5 butonuna basıldığında ise azaltan programın yazılması.(PortD'nin 3. bitine bağlı ledi program çalıştığı sürece yakarsak ledlerin parlaklıklarındaki değişimi izlenebilir)



```
            LIST        P16F877
            INCLUDE     "P16F877.INC"
IS          EQU         H '21'          ; İş süresi
CEVRIM      EQU         H '22'          ; Çevrim süresi=iş süresi+ bekleme süresi
ISGEC       EQU         H '23'          ; İş süresinin geçici olarak saklandığı yazmac
SAYAC1      EQU         H '24'          ; gecikme altprogram kayıtçıları
SAYAC2      EQU         H '25'          ;  "       "
            ORG         0x003           ; Reset vektörü
            GOTO        ILK_DEGER
GECIKME
            MOVLW       h '0F'
            MOVWF       SAYAC1
DONGU1
            MOVLW       h'FF'
            MOVWF       SAYAC2
DONGU2
            DECFSZ      SAYAC2,F
            GOTO        DONGU2
            DECFSZ      SAYAC1,F
            GOTO  DONGU1
            RETURN
ILK_DEGER
            CLRF        PORTD
            CLRF        PORTB
            BCF         STATUS,RP1
            BSF         STATUS,RP0
            BCF         OPTION_REG, NOT_RBPU ; pull-up dirençleri aktif
            MOVLW       B '00110000'            ; PortB'ye butonlar bağlı
            MOVWF       TRISB
            CLRF        TRISD
            BCF         STATUS, RP0
BASLA
            MOVLW       D'25'
            MOVWF       IS              ; iş çevrim süresi=25
            MOVWF       ISGEC           ;
            BSF         PORTD,2         ; iş süresi= bekleme süresi, kare dalga
DONGU
            MOVF        ISGEC,W
            MOVWF       IS
            MOVLW       D'250'
            MOVWF       CEVRIM          ; Bir peryot 250 birim
            BSF         PORTD,0
            BTFSS       PORTB,4         ; B4 tuşuna basıldımı?
            GOTO        ARTTIR_IS       ;İş süresini arttırma altprogramına git
            BTFSS       PORTB,5         ; B5 tuşuna basıldımı?
            GOTO        AZALT_IS        ; İş süresini azaltmaya git
KONTROL_IS
            DECFSZ      IS,F            ; IS süresini 1 azalt, 0 mı?
            GOTO        KONTROL_CEVRIM
            BCF         PORTD,0         ;
KONTROL_CEVRIM
            DECFSZ      CEVRIM,F        ;Çevrimi 1 azalt 0 mı?
            GOTO        KONTROL_IS ; Çevrim sıfırlanmadı iş süresine bak
            GOTO  DONGU
ARTTIR_IS
            CALL        GECIKME
            MOVF        ISGEC,W
            MOVWF       IS
            INCF        IS,F
```

```
            MOVF    IS,W
            MOVWF   ISGEC
            GOTO    DONGU
AZALT_IS
            CALL    GECIKME
            MOVF    ISGEC,W
            MOVWF   IS
            DECF    IS,F
            MOVF    IS,W
            MOVWF   ISGEC
            GOTO    DONGU
            END
```

9.8.2. A/D Dönüşümü

PIC 16F877'de analog giriş için E portunda 3 ve A portunda 5 pin bulunmaktadır. Bu uçlara bağlanacak olan sıcaklık, ışık, ses veya sensörler yardımıyla analog veri alınabilir. AD dönüştürücü uyuma modunda da çalışabilir. Kendi iç devre saatini için ise RC osilatörünü kullanır. AD çevrim işlemleri için 4 adet yazmaç kullanılır. Bu yazmaçlar; ADRESH, ADRESL, ADCON0 ve ADCON1'dir. Bu yazmaçlardan ADRESH ve ADRESL , AD dönüşüm sonucunun üst ve alt bytelarını tutulduğu kısımdır. Diğer yazmaçlar ise kontrol yazmaçlarıdır. ADCON0, AD çevirme işlemlerinin kontrolün de, ADCON1 ise port pinlerinin konfigürasyonlarında kullanılır.

ADCON0 Yazmacı

7	6	5	4	3	2	1	0
ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/ DONE	-	ADON

00: $F_{OSC}/2$
01: $F_{OSC}/8$
10: $F_{OSC}/32$
11: F_{RC}

Analog kanal seçim bitleri 5:3

1: dönüşüm başladı
0:Bitti

1:AD çalışıyor
0: Kapalı

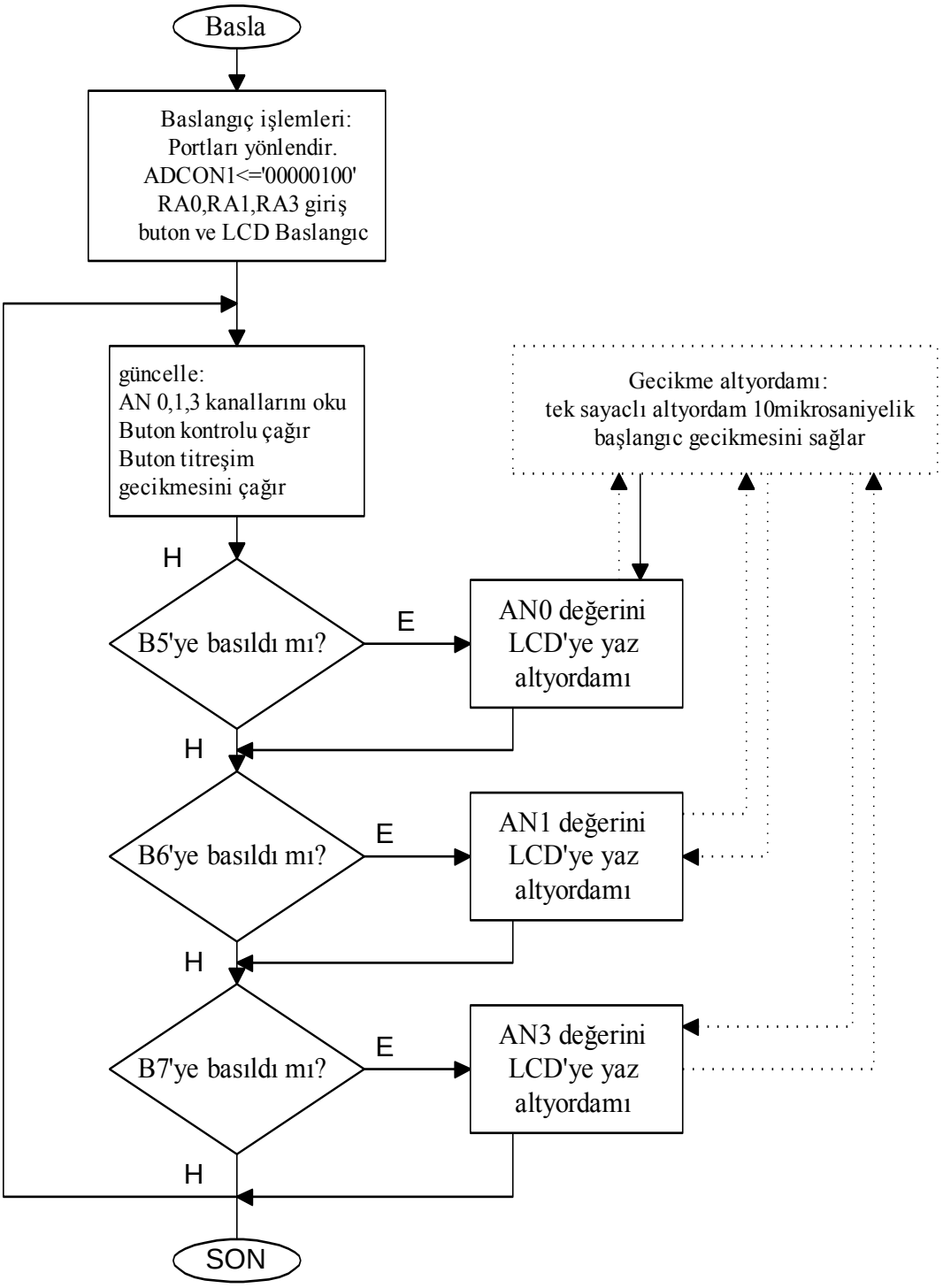
ADCON1 Yazmacı

7	6	5	4	3	2	1	0
ADFM				PCFG3	PCFG2	PCFG1	PCFG0

AD sonuç seçim biti
0: Sağa yanaşık
1: Sola Yanaşık

AD Port konfigürasyon bitleri

Örnek 9.18 : PIC16F877 kullanarak, PORRTA'nın 0,1 ve 3. bitlerine bağlı 3 potansiyometrenin direncinin sayısal değerini (AN0, AN1, AN3) butonlarına basıldığında PORTD'ye bağlı LCD biriminde gösteren programın yazılması.



PortA daki potansiyometrelerden (A0,A1 ve A3) analog değerler düzenli olarak okunur, 8 bit çözünürlükte sayısal değere dönüştürülerek saklanır.

B1,B2, ve B3 butonlarından birine basılması ile o butona karşılık gelen değerde (A0,A1ve A3) okunup AN0,AN1,AN3 de saklanan değerleri yine 8 bit olarak PORTD‘ ye bağlı LCD de gösterilmektedir. Butona her yeni basışta o butona karşılık gelen değerin en güncel hali LCD’de gösterilir. Butonların birine yeniden basıncaya kadar LCD de gösterilir.

```
        Bu deney PIC lab kartında denenmek üzere hazırlanmıştır.
*****************************************************************
        Title "A/D çevrim uygulaması"
        list p=16f877
#include <P16F877.INC>;MPASM standart değişken tanımları
_CONFIG(_CP_OFF&WDT_ON&_PWRTE_OFF&_RC_OSC)
        errolevel-302      ;ignore error when storing to bank 1
*****************************************************************
                                        Değişken tanımları
*****************************************************************
variable KRAM=0x03              CBLOK…ENDC
cblock KRAM
AN0                     ;Analog değişkenler     *AN0 EQU '20'
AN1                                                *ANI  EQU  '21'
AN3                                                *AN3  EQU'22'
SAYAC
Endc
Variable KRAM=SAYAC+1
*****************************************************************
org     0x03            ;program başlangıç adresi
goto   Basla

*****************************************************************
Altprogramlar ve altprogram kitaplıkları
#inlude <PBS.INC>; Buton kontrol rutini
#inlude<B2D.INC>;
#include<LCD.INC>;

*****************************************************************
        A/D dönüştürmede dahili RC saati kullanılır. Okunan her kanal kendisine ait bir yazmaçta saklanmaktadır. A0
kanalı yani RA0 bacağında okunan analog değer 8 bit olarak AN0  yazmacında saklanmaktadır.Aynı şekilde RA1
bacağındaki değer AN1 , RA3 'deki değerse AN3 yazmacında saklanır.
*****************************************************************

AN0_KanalınıOku
        Movlw       b'11000001'     ; RC osilatör ve A0 kanalının seçilmesi.
        movwf       ADCON0
        call        Gecikme
        bsf         ADCON0,GO       ; A/D dönüştürme işlemini başlatan bit.
        btfsc       ADCON0,GO
        goto        $-1
        movf        ADRESH,W
        movwf       AN0
        return

 AN1_KanalınıOku
        movlw       b'11001001'     ; RC osilatör ve A0 kanalının seçilmesi.
        movwf       ADCON0
        call        Gecikme
        bsf         ADCON0,GO       ;A/D dönüştürme işlemini başlatılmaktadır
        btfsc       ADCON0,GO
        goto        $-1
        movf        ADRESH,W
        movwf       AN1
        return

AN3_KanalınıOku
        movlw       b'11011001'     ; RC osilatör ve A0 kanalının seçilmesi.
        movwf       ADCON0          ; birlikte seçilmesi
        call        Gecikme
        bsf         ADCON0,GO       ; A/D dönüştürme işlemini başlatılmakta.
        btfsc       ADCON0,GO
        goto        $-1
        movf        ADRESH,W
        movwf       AN03
```

```
        return

AN0_değerlerini LCD ye yaz
        Movf        AN0,W
        Call        B2D
        Movlw       0x30
        Addwf       BIRLER,1
        Addwf       ONLAR,1
        Addwf       YÜZLER,1
        Call        LCD Sıfırla
        Movlw'A'
        Call        LCDyeKarakterGonder
        Movlw'N'
        Call        LCDyeKarakterGonder
        Movlw'.'
        Call        LCDyeKarakterGonder
        Movlw'0'
        Call        LCDyeKarakterGonder
        Movlw' '
        Call        LCDyeKarakterGonder
        Movlw'='
        Call        LCDyeKarakterGonder
        Movlw' '
        Call        LCDyeKarakterGonder
        Movf        YUZLER,0
        Call        LCDyeKarakterGonder
        Movf        ONLAR,0
        Call        LCDyeKarakterGonder
        Movf        BİRLER,0
        Call        LCDyeKarakterGonder
        Return

AN1_değerlerini LCD ye yaz
        Movf        AN1,W
        Call        B2D
        Movlw       0x30
        Addwf       BIRLER,1
        Addwf       ONLAR,1
        Addwf       YÜZLER,1
        Call        LCDSıfırla
        Movlw'A'
        Call        LCDyeKarakterGonder
        Movlw'N'
        Call        LCDyeKarakterGonder
        Movlw'.'
        Call        LCDyeKarakterGonder
        Movlw'1'
        Call        LCDyeKarakterGonder
        Movlw' '
        Call        LCDyeKarakterGonder
        Movlw'='
        Call        LCDyeKarakterGonder
        Movlw' '
        Call        LCDyeKarakterGonder
        Movf        YUZLER,0
        Call        LCDyeKarakterGonder
        Movf        ONLAR,0
        Call        LCDyeKarakterGonder
        Movf        BİRLER,0
        Call        LCDyeKarakterGonder
        Return

AN3_değerlerini LCD ye yaz
        Movf        AN3,W
        Call        B2D
```

```
        Movlw       0x30
        Addwf       BIRLER,1
        Addwf       ONLAR,1
        Addwf       YÜZLER,1
        Call        LCD Sıfırla
        Movlw'A'
        Call        LCDyeKarakterGonder
        Movlw'N'
        Call        LCDyeKarakterGonder
        Movlw'.'
        Call        LCDyeKarakterGonder
        Movlw'3'
        Call        LCDyeKarakterGonder
        Movlw' '
        Call        LCDyeKarakterGonder
        Movlw'='
        Call        LCDyeKarakterGonder
        Movlw' '
        Call        LCDyeKarakterGonder
        Movf        YUZLER,0
        Call        LCDyeKarakterGonder
        Movf        ONLAR,0
        Call        LCDyeKarakterGonder
        Movf        BİRLER,0
        Call        LCDyeKarakterGonder
        Return
```

****

"Gecikme" rutini A/D işleminin başlayabilmesi için gereken ve yazılımla sağlanan yaklaşık 10 us. lik gecikme sağlayan rutindir. 4 MHZ 'lik bir saat osilatöründe aşağıdaki döngü 3us.almaktadır .Eğer "SAYAC" değerine başlangıç değeri olarak 3 atanırsa , toplamda 10 us.den biraz daha büyük bir değere ulaşan gecikme süresi elde edilir.

****

```
Gecikme
        Movlw       0x03        ;SAYAC 'a 3 değerini ata.
        Movwf       SAYAC
        Decfsz      SAYAC,F     ;Gecikme döngüsü
        Goto        $-1
        Return
**********************************************************************
                                Ana Program
**********************************************************************
Basla
        Movlw  0xFF             ;PORTD'nin tüm bitleri"1"
        Movwf  PORTD
        Bsf         STATUS,5    ;bank1
        Movwf  TRISA  ; PortAnın tüm bitleri giriş
        Clrf        TRISD  ; PortD nin tüm bitleri çıkış
        Movlw  b '00000100'     ; RA0,RA1,RA3 analog giriş
        Movwf       ADCON1
        Bcf         STATUS,5    ; bank0
        Call        ButonKontrolBaslangic
        Call        LCDBaslangic
Guncelle
        Call        AN0_KanaliniOku
        Call        AN1_KanaliniOku
        Call        AN2_KanaliniOku
        Call        ButonKontrol
        Call        ButonTitresimGecikmesi
        Btfsc       PB1
        Call        AN0_DegeriniLCDyeYaz
        Btfsc       PB2
        Call        AN1_DegeriniLCDyeYaz
        Btfsc       PB3
        Call        AN3_DegeriniLCDyeYaz
        Goto        Guncelle
```

```
        End
```
Makro dosyalarının listeleri:
BD2.INC dosyasının listesi
```
--------------------------------B2D.INC-------------------------------
        cblock   KRAM
        SAYI                            ; w kayıtçısındaki sayının kopyası
        YUZLER                          ; Yuzler basamağı
        ONLAR                   ; Onlar basamağı
        BIRLER                          ; Birler basamağı
        Endc
        VariableKRAM=BIRLER+1

B2D     movwf   SAYI
        Clrf            YUZLER
        Clrf            ONLAR
        Clrf            BIRLER
YuzB
        Movlw   .100
        Subwf           SAYI,W
        Btfsc           STATUS,C
        Goto            YuzlerBasamagı
OnB
        Movlw           .10
        Subwf           SAYI,W
        Btfsc           STATUS,C
        Goto            OnlarBasamagı
        Movf            SAYI,W
        Movwf           BIRLER
        Return

YuzlerBasamagi
        İncf            YUZLER,1
        Movwf           SAYI
        Goto            YuzB
OnlarBasamagi
        İncf            ONLAR,1
        Movwf           SAYI
        Goto            OnB

------------------------------PBS.INC dosyası---------------------------------

TUŞ TAKIMI

        Cblock  KRAM
        BUTON18
        BUTON916
        SAYAC1
        SAYAC2
        Endc
        Variable KRAM=SAYAC2+1
#define         PB1             BUTON18,0
#define         PB2             BUTON18,1
#define         PB3             BUTON18,2
#define         PB4             BUTON18,3
#define         PB5             BUTON18,4
#define         PB6             BUTON18,5
#define         PB7             BUTON18,6
#define         PB8             BUTON18,7
#define         PB9             BUTON916,0
#define         PB10            BUTON916,1
#define         PB11            BUTON916,2
#define         PB12            BUTON916,3
#define         PB13            BUTON916,4
#define         PB14            BUTON916,5
#define         PB15            BUTON916,6
```

```
#define         PB16            BUTON916,7

ButonKontrolBaslangic
        Clrf            BUTON18  ; PB1..8 arası tuş değerleri BUTON18 yazmacının 0..7  bitlerinde,

        Clrf            BUTON916;
        Movlw  0xFF
        Movwf  PORTB
        Bsf             STATUS,5
        Bsf             OPTION_REG,7 ; PortB çekme dirençleri devrede
        Movlw           0xF0
        Movwf           TRISB
        Bcf             STATUS,5
        Return

ButonKontrol
; PB1..4 arası tuş kontrolu

Ts1     movlw           b'11111110'
        Movwf           PORTB
        Btfss           PORTB,4
        Goto            Tus1
        Bcf             PB1
Ts2     Btfss           PORTB,5
        Goto            Tus2
        Bcf             PB2
Ts3     Btfss           PORTB,6
        Goto            Tus3
        Bcf             PB3
Ts4     Btfss           PORTB,7
        Goto            Tus4
        Bcf             PB4

;PB5..8 arası tuş kontrolu
Ts5     movlw           b'11111101'
        Movwf           PORTB
        Btfss           PORTB,4
        Goto            Tus5
        Bcf             PB5
Ts6     Btfss           PORTB,5
        Goto            Tus6
        Bcf             PB6
Ts7     Btfss           PORTB,6
        Goto            Tus7
        Bcf             PB7
Ts8     Btfss           PORTB,7
        Goto            Tus8
        Bcf             PB8

;PB9..12 arası tuş kontrolu
Ts9     movlw           b'11111011'
        Movwf           PORTB
        Btfss           PORTB,4
        Goto            Tus9
        Bcf             PB9
Ts10    Btfss           PORTB,5
        Goto            Tus10
        Bcf             PB10
Ts11    Btfss           PORTB,6
        Goto            Tus11
        Bcf             PB11
Ts12    Btfss           PORTB,7
        Goto            Tus12
        Bcf             PB12
```

```
;PB13..16 arası tuş kontrolu
Ts13    movlw       b'11110111'
        Movwf       PORTB
        Btfss       PORTB,4
        Goto        Tus13
        Bcf         PB13
Ts14    Btfss       PORTB,5
        Goto        Tus14
        Bcf         PB14
Ts15    Btfss       PORTB,6
        Goto        Tus15
        Bcf         PB15
Ts16    Btfss       PORTB,7
        Goto        Tus16
        Bcf         PB16
        Return

Tus1
        Bsf         PB1
        Goto        Ts2
Tus2
        Bsf         PB2
        Goto        Ts3
Tus3
        Bsf         PB3
        Goto        Ts4
Tus4
        Bsf         PB4
        Goto        Ts5
Tus5
        Bsf         PB5
        Goto        Ts6
Tus6
        Bsf         PB6
        Goto        Ts7
Tus7
        Bsf         PB7
        Goto        Ts8
Tus8
        Bsf         PB8
        Goto        Ts9
Tus9
        Bsf         PB9
        Goto        Ts10
Tus10
        Bsf         PB10
        Goto        Ts11
Tus11
        Bsf         PB11
        Goto        Ts12
Tus12
        Bsf         PB12
        Goto        Ts13
Tus13
        Bsf         PB13
        Goto        Ts14
Tus14
        Bsf         PB14
        Goto        Ts15
Tus15
        Bsf         PB15
        Goto        Ts16
Tus16
        Bsf         PB16
        Return
```

```
ButonTitresimGecikmesi
        Movlw       0x40
        Movwf       SAYAC1
Yukle   Movlw       0xFF
        Movwf  SAYAC2
Azalt   Decfsz      SAYAC2,F
        Goto        Azalt
        Decfsz      SAYAC1,F
        Goto        Yukle
        Return
```

LCD.INC dosyasının listesi;

```
********** LCD.INC **************************
; değişkenler
cblock          KRAM
SAY1
SAY2
Endc
Variable        KRAM=SAY2+1

#define  RS     PORTE,0
#define  EN     PORTE,1
#define  RW     PORTE,2

LCDBaslangic
        Bsf         STATUS,RP0
        Movlw       b'00000010'
        Movwf       ADCON1      ;A portunun RA4 hariç, bitleri analog giriş
        Movlw       b'00000000'
        Movwf       TRISD
        Movwf       TRISE
        Bcf         STATUS,RP0
        Bcf         RW          ; LCD'ye yazma işlemi
        Bcf         EN          ;
        Bcf         RS          ;
        Call        _125us_gecikme ; 125 mikrosaniye gecikme
        Movlw  0x38             ;8 bit 5x7
        Movwf       PORTD ; 00111000
        Call        Darbe       ;
        Movlw       0x0F
        Movwf       PORTD ; 0000 1111
        Call        Darbe
        Movlw       0x01        ; göstergeyi temizler
        Movwf       PORTD ; 0000 0001
        Call        Darbe
        Call        _5ms_gecıkme
        Return

LCDyeKarakterGonder
        Movwf       PORTD
        Bcf         RW
        Bsf         RS
        Call        Darbe
        Return

LCDsıfırla
        Bcf         RW
        Bcf         EN
        Bcf         RS
        Call        _125us_gecikme
        Movlw       0x01
        Movwf       PORTD
        Call        Darbe
        Call        _5ms_gecikme
```

```
        Return

_125us_gecikme ; Ortalama 42*3=126 çevrim elde etmektedir.
        Movlw       0x2A
        Movwf       SAY1
Gec1    decfsz      SAY1,f
        Goto        Gec1
        Return

_5ms_gecikme
        movlw       0x29
        movwf       SAY2
Gec2    call        _125us_gecikme
        Decfsz      SAY2,F
        Goto        Gec2
        Return

Darbe
        Bsf         EN
        Nop
        Bcf         EN
        Call        _125us_gecikme
        Return
```

9.9. USART

PIC 16F877 de kullanılan ilk seri giriş ve çıkış birimi usart olarak adlandırılmaktadır. Bu arabirim CRT terminaller, PC'ler, çevre birimler, seri EPROM gibi birimlerle iletişime geçer.

USART biriminin iki önemli yazmacı; RSCTA ve Baud hız kaynağı SPBRG yazmacıdır.SPBRG 8 bitlik zamanlayıcıdır.Veri tablolarını incelenerek gerekli baud hızı seçilir.Asenkron çalıştırılacaksa BRGH(TXSTA<2>) biti de baud hızlarını kontrol edebilir.BRGH senkron moda gerekmez.Baud hızı ile ilgili üç tane yazmaç vardır. Bunlar ; TXSTA, RCSTA, SPBRG' dir.

USART asenkron modunda ençok kullanılan biçimler;standart 1 başlama(start) biti , 8 veya 9 veri biti, 1 bitiş(stop) bitidir.USART modülü veri iletişiminde önce en düşük öncelikli biti seçer. Parite biti donanım tarafından yakalanamaz, ancak istendiği takdirde yazılım tarafından yakalanabilir. USART uyuma (sleep) modunda çalışmaz. USART'ın 4 önemli elemanı vardır. Bunlar Baud hız kaynağı, Örnekleme devresi, Asenkron iletim ve Asenkron alıcıdır. Asenkron iletim yazmaçları ise şunlardır. PIR, RCSTA, TXREG, PIE, TXTA, SPBRG'dir.

Aşağıda yazılan bu program ile asenkron modda seri iletişimin kurulmasını sağlar.

```
LIST            P16F877
INCLUDE         "P16F877.INC"
Variable        KRAM=0x20
cblock          KRAM
SAYAC
Endc
 Variable       KRAM=SAYAC+1
ORG             0X03
GOTO   BASLANGIC
```

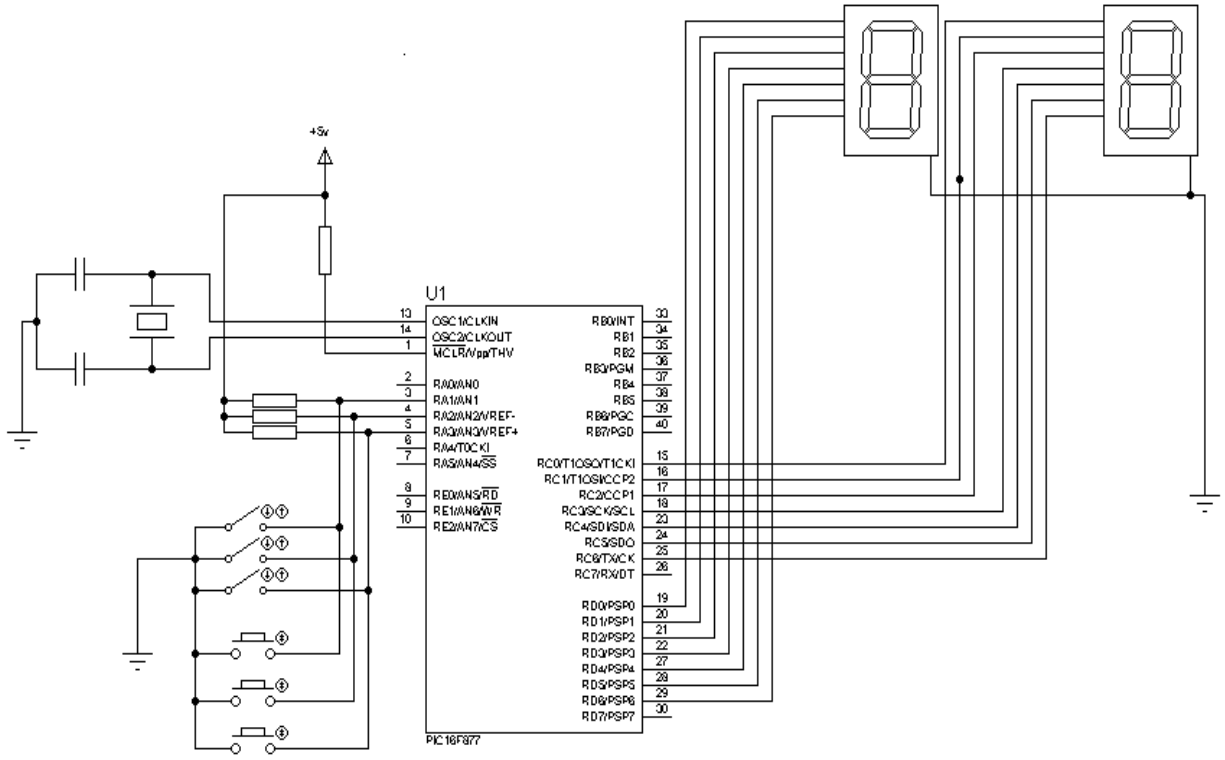
```
                ORG             0X05
BASLANGIC
                BSF             STATUS,RP0
                MOVLW           h ‘19’
                MOVWF           SPBRG
                MOVLW           b ‘00100100’
                MOVWF           TXSTA
                BCF             STATUS,RP0
                MOVLW           b ‘10010000’
                MOVWF           RCSTA
                CALL            LCDBASLANGIC
                CALL            LCDSIFIRLA
ANADONGU
                CALL            VERİAL
                MOVWF           TXREG
                CALL            LCDye karakter gönder
                GOTO            ANADONGU
VERİAL
                MOVLW           h ‘06’
                ANDWF           RCSTA,W
                BTFSS           STATUS,Z
                GOTO            VERİALIMIHATALI
VERİHAZIRMI
                BTFSS           PIR1,5
                GOTO            VERİHAZIRMI
                MOVF            RCREG,W
                BCF             PIR1,5
                RETURN
VERİALIMIHATALI
                BCF             RCSTA,4
                BSF             RCSTA,4
                GOTO            VERİAL
                İNCLUDE         <LCD.İNC>
                END
```

9.10. 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. *Y. Bodur, 2001. “Adım Adım PICmicro Programlama”,İnfogate.*
4. *Ç. Akpolat, 2005.“PIC Programlama”, Pusula Yayıncılık*

BÖLÜM 10. UYGULAMALAR

10.1. İleri Geri Sayıcı Devresi




```
;İLERİ GERİ SAYICI (0--99)
        list    p=16f877
        INCLUDE "P16F877.INC"
        __CONFIG(_WDT_OFF&_XT_OSC&_PWRTE_ON&_CP_OFF)
S1              EQU             H'20'
S2              EQU             H'21'
BIRL            EQU             H'22'
ONL             EQU             H'23'
UMUMI           EQU             H'24'
                ORG             H'000'
                GOTO            BASLA
BASLA
                BSF             STATUS,5
                CLRF            TRISD
                CLRF            TRISC
                MOVLW           B'00001110'
                MOVWF           TRISA
                MOVLW           H'07'
                MOVWF           ADCON1          ;Port A-E  Sayısal Giriş Olarak  Düzenleniyor..
                BCF             STATUS,5
                CLRF            PORTB
                CLRF            PORTC
                COMF            PORTB,F
                COMF            PORTC,F
                CLRF            ONL
                CLRF            BIRL
                GOTO            GORUNTULE
BUTONDENETIM
                BTFSS           PORTA,1
                GOTO            ILERI
                BTFSS           PORTA,2
                GOTO            GERI
                BTFSS           PORTA,3
                GOTO            SIFIR
```

```
                GOTO            BUTONDENETIM
ILERI
                INCF            BIRL,F
                MOVF            BIRL,W
                XORLW           H'0A'
                BTFSS           STATUS,Z
                GOTO            GORUNTULE
                CLRF            BIRL
                INCF            ONL,F
                MOVF            ONL,W
                XORLW           H'0A'
                BTFSS           STATUS,Z
                GOTO            GORUNTULE
                CLRF            ONL
                CLRF            BIRL
                GOTO            GORUNTULE
GERI
                DECF            BIRL,F
                MOVF            BIRL,W
                XORLW           H'FF'
                BTFSS           STATUS,Z
                GOTO            DEVAM
                MOVLW           H'09'
                MOVWF           BIRL
                DECF            ONL,F
                MOVF            ONL,W
                XORLW           H'FF'
                BTFSS           STATUS,Z
                GOTO            DEVAM
                MOVLW           H'09'
                MOVWF           ONL
DEVAM
                GOTO            GORUNTULE
SIFIR
                CLRF            ONL
                CLRF            BIRL
                GOTO            GORUNTULE
GORUNTULE
                MOVF            BIRL,W
                CALL            TABLO
;-------------------------------------
;               MOVWF           UMUMI
;               COMF            UMUMI,W
;-------------------------------------
                MOVWF           PORTC           ;BIRLERI  GORUNTULE....

                MOVF            ONL,W
                CALL            TABLO
;-------------------------------------
;               MOVWF           UMUMI
;               COMF            UMUMI,W
;-------------------------------------
                MOVWF           PORTD           ;ONLARI  GORUNTULE....
TEST
                CALL            BEKLE_15
                MOVF            PORTA,W
                ANDLW           B'00001110'
                XORLW           B'00001110'
                BTFSS  STATUS,Z
                GOTO            TEST
                GOTO            BUTONDENETIM
TABLO
                ADDWF           PCL,F
                RETLW           H'3F'    ;--0--
                RETLW           H'06'
```

```
                RETLW       H'5B'
                RETLW       H'4F'
                RETLW       H'66'
                RETLW       H'6D'
                RETLW       H'7D'
                RETLW       H'07'
                RETLW       H'7F'
                RETLW       H'6F'    ;--9--
BEKLE_15
                MOVLW       H'2A'    ;H'1A'
                MOVWF       S1
DON1
                MOVLW       H'FF'    ;H'FF'
                MOVWF       S2
DON2
                DECFSZ      S2,F
                GOTO        DON2
                DECFSZ      S1,F
                GOTO        DON1
                RETURN
                END
```

10.2. Üç Kademeli Dc Motor Kontrolü

3 tane anahtar yardımıyla elimizdeki dc motoru yavaş , hızlı ve daha hızlı olarak kontrol edebiliriz. İlk bakışta normal bir vantilatör görünümünde olmasına rağmen , en büyük fark kullanılan anahtarların analog olmaması. Yani anahtara basıldığında analog olarak gerilim değişmesi sağlamak yerine, anahtara dokunmak suretiyle dijital bir elektrik değişimi sağlanmaktadır.

```
          include  "p16f877.inc"
NUL             EQU             020H
BEK1            EQU             021H
BEK2            EQU             022H
BEK3            EQU             023H
SAYI            EQU             024H
SAYI1           EQU             025H
SAYI2           EQU             026H
SAYI3           EQU             027H
STATE  EQU              028H
REG1            EQU             029H
REG2            EQU             02AH
REG3            EQU             02BH
DEGER EQU               02CH
                ORG             0H
                GOTO            START
                ORG             4
                GOTO            INTRR
                ORG             5
start
                BSF             STATUS,RP0
                MOVLW           07H
                MOVWF           ADCON1
                MOVLW           0FFH
                MOVWF           TRISA
                CLRF            TRISD
                CLRF            TRISB
                CLRF            TRISE
                BCF             TRISC,2
                MOVLW           0FFH
                MOVWF           PR2
```

```
        MOVLW   030H
        MOVWF   OPTION_REG
        BSF     PIE1,TMR1IE
        BSF     INTCON,GIE
        BSF     INTCON,PEIE
        BCF     STATUS,RP0
        BCF     PIR1,TMR1IF
        MOVLW   00H
        MOVWF   STATE
        CLRF    TMR0
        CALL    TIMER1
        CLRF    PORTA
        CLRF    PORTB
        CLRF    PORTC
        CLRF    PORTD
        CLRF    PORTE
        CLRF    CCP1CON
        CLRF    TMR2
        MOVLW   0H
        MOVWF   CCPR1L
        MOVLW   03CH
        MOVWF   CCP1CON
        BSF     T2CON,2
        BSF     T2CON,1
        BSF     INTCON,7
        MOVLW   038H
        CALL    LCD
        MOVLW   0FH
        CALL    LCD
        MOVLW   01H
        CALL    LCD
OKU
        BSF     STATUS,RP0
        BSF     PIE1,TMR1IE
        BCF     STATUS,RP0
        CALL    TUSOKU
        MOVWF   NUL
        CLRW
        SUBWF NUL,W
        BTFSC  STATUS,Z
        GOTO    OKU
        MOVLW   '#'
        SUBWF NUL,W
        BTFSC  STATUS,Z
        GOTO    OKU
        MOVLW   '*'
        SUBWF NUL,W
        BTFSC  STATUS,Z
        GOTO    OKU
        BSF     STATUS,RP0
        BCF     PIE1,TMR1IE
        BCF     STATUS,RP0
        CLRF    REG3
        CLRF    REG2
        CLRF    REG1
        BCF     PORTE,1
        MOVLW   080H
        CALL    LCD
        BSF     PORTE,1
        MOVF    NUL,W
        MOVWF   REG3
        CALL    LCD
OKU2
        BCF     PORTE,1
        MOVLW   081H
```

```
                CALL    LCD
                BSF     PORTE,1
                MOVLW   ' '
                CALL    LCD
                CALL    TUSOKU
                MOVWF   NUL
                CLRW
                SUBWF   NUL,W
                BTFSC   STATUS,Z
                GOTO    OKU2
                MOVLW   '#'
                SUBWF   NUL,W
                BTFSC   STATUS,Z
                GOTO    OKU
                MOVLW   '*'
                SUBWF   NUL,W
                BTFSC   STATUS,Z
                GOTO    OKU
                BCF     PORTE,1
                MOVLW   081H
                CALL    LCD
                BSF     PORTE,1
                MOVF    NUL,W
                MOVWF   REG2
                CALL    LCD
OKU3
                BCF     PORTE,1
                MOVLW   082H
                CALL    LCD
                BSF     PORTE,1
                MOVLW   ' '
                CALL    LCD
                CALL    TUSOKU
                MOVWF   NUL
                CLRW
                SUBWF   NUL,W
                BTFSC   STATUS,Z
                GOTO    OKU3
                MOVLW   '#'
                SUBWF   NUL,W
                BTFSC   STATUS,Z
                GOTO    OKU
                MOVLW   '*'
                SUBWF   NUL,W
                BTFSC   STATUS,Z
                GOTO    OKU2
                BCF     PORTE,1
                MOVLW   082H
                CALL    LCD
                BSF     PORTE,1
                MOVF    NUL,W
                MOVWF   REG1
                CALL    LCD
OKU4
                CALL    TUSOKU
                MOVWF   NUL
                CLRW
                SUBWF   NUL,W
                BTFSC   STATUS,Z
                GOTO    OKU4
                MOVLW   '#'
                SUBWF   NUL,W
                BTFSC   STATUS,Z
                GOTO    KONTROL
                MOVLW   '*'
```

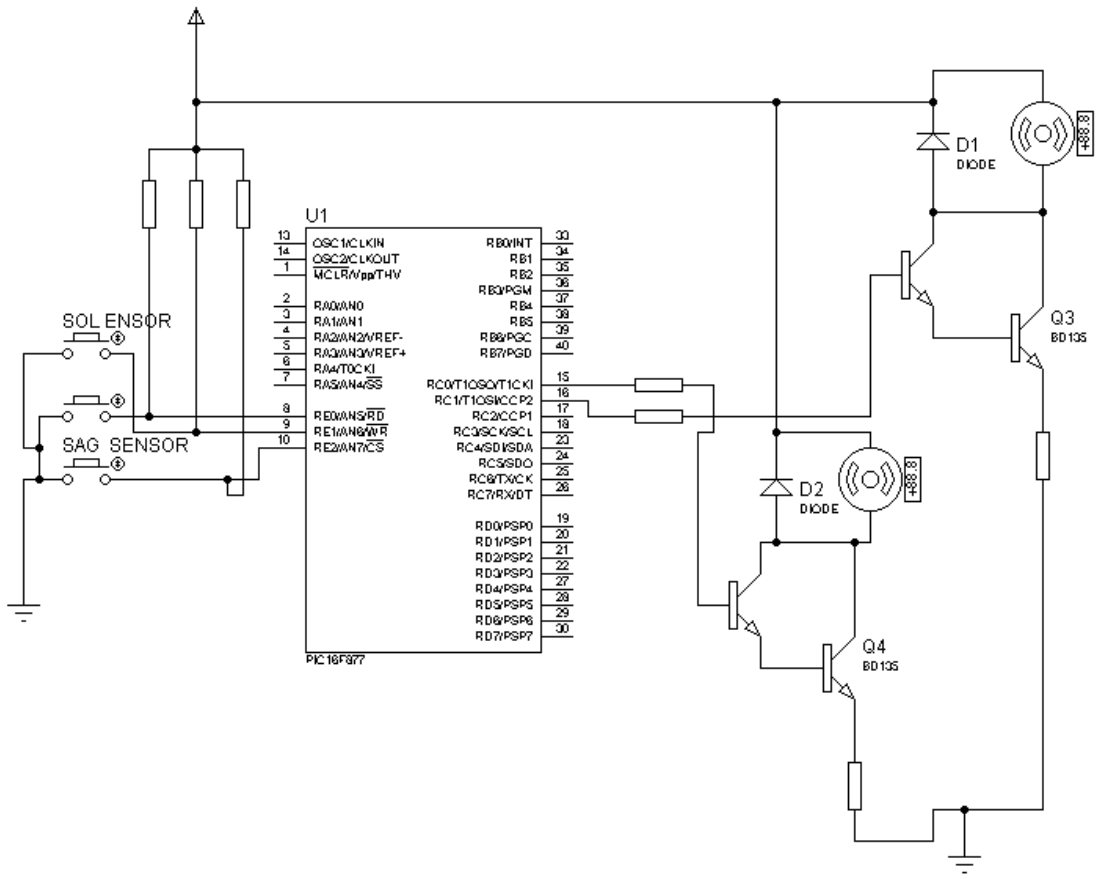
```
            SUBWF NUL,W
            BTFSC STATUS,Z
            GOTO  OKU3
            GOTO  OKU4
KONTROL
            MOVLW 0FH
            ANDWF REG1,F
            ANDWF REG2,F
            ANDWF REG3,F
            CLRF  DEGER
            MOVLW 064H
            MOVWF NUL
TOP1
            MOVF  REG3,W
            ADDWF DEGER,F
            DECFSZ NUL,F
            GOTO  TOP1
            MOVLW 0AH
            MOVWF NUL
TOP2
            MOVF  REG2,W
            ADDWF DEGER,F
            DECFSZ NUL,F
            GOTO  TOP2
            MOVF  REG1,W
            ADDWF DEGER,F
            MOVF  DEGER,W
            MOVWF CCPR1L
            GOTO  OKU
INTRR
            BTFSS PIR1,TMR1IF
            RETFIE
            BCF   PIR1,TMR1IF
            CALL  TIMER1
            INCF  STATE,F
            BTFSC STATE,0
            RETFIE
            MOVF  TMR0,W
            MOVWF SAYI
            CLRF  TMR0
            BSF   STATUS,RP0
            MOVLW 030H
            MOVWF OPTION_REG
            BCF   STATUS,RP0
            BCF   PORTE,1
            MOVLW 01H
            CALL  LCD
            MOVLW 086H
            CALL  LCD
            CLRF  SAYI1
            CLRF  SAYI2
            CLRF  SAYI3
            CLRW
            SUBWF SAYI,W
            BTFSC STATUS,Z
            GOTO  YAZ
CIKAR
            MOVLW 0AH
            SUBWF SAYI,F
            INCF  SAYI2,F
            BTFSS STATUS,C
            GOTO  SON
            MOVLW 0AH
            SUBWF SAYI2,W
            BTFSS STATUS,Z
```

```
        GOTO    CIKAR
        CLRF    SAYI2
        INCF    SAYI3,F
        GOTO    CIKAR
SON
        DECF    SAYI2,F
        MOVLW   0AH
        ADDWF   SAYI,F
        MOVF    SAYI,W
        MOVWF   SAYI1
YAZ
        BSF     PORTE,1
        MOVF    SAYI3,W
        IORLW 030H
        CALL    LCD
        MOVF    SAYI2,W
        IORLW 030H
        CALL    LCD
        MOVF    SAYI1,W
        IORLW 030H
        CALL    LCD
        BCF     PORTE,0
        MOVLW   081H
        CALL    LCD
        RETFIE
TIMER1
        BCF     T1CON,0
        MOVLW   0BH
        MOVWF   TMR1H
        MOVLW   0DBH
        MOVWF   TMR1L
        MOVLW   039H
        MOVWF   T1CON
        RETURN
LCD
        MOVWF   PORTB
        BSF     PORTE,0
        MOVLW   03H
        MOVWF   BEK1
TEKRAR
        MOVLW   0FFH
        MOVWF   BEK2
        DECFSZ  BEK2,F
        GOTO    $-1
        DECFSZ  BEK1,F
        GOTO    TEKRAR
        BCF     PORTE,0
        RETURN
        ORG     0100H
TUSOKU
        BSF     PCLATH,0
        CALL    TUSGECIK
        CLRF    PORTA
        MOVLW   B'00000111'
        MOVWF   PORTD
        CALL    GECIK1
        BTFSC  PORTA,1
        GOTO    SIRA2
        BTFSC  PORTA,2
        GOTO    SIRA3
        BTFSC  PORTA,0
        GOTO    SIRA1
        BTFSC  PORTA,3
        GOTO    SIRA4
        CLRF    PORTD
```

```
        RETLW 0H
SIRA1
        BSF     PORTD,2
        BCF     PORTD,1
        BCF     PORTD,0
        BTFSC  PORTA,0
        RETLW'1'
        BCF     PORTD,2
        BCF     PORTD,1
        BCF     PORTD,0
        BTFSC  PORTA,0
        RETLW'2'
        BCF     PORTD,2
        BCF     PORTD,1
        BSF     PORTD,0
        BTFSC  PORTA,0
        RETLW'3'
        GOTO    TUSOKU
SIRA2
        BSF     PORTD,2
        BCF     PORTD,1
        BCF     PORTD,0
        BTFSC  PORTA,1
        RETLW'4'
        BCF     PORTD,2
        BCF     PORTD,1
        BCF     PORTD,0
        BTFSC  PORTA,1
        RETLW'5'
        BCF     PORTD,2
        BCF     PORTD,1
        BSF     PORTD,0
        BTFSC  PORTA,1
        RETLW'6'
        GOTO    TUSOKU
        SIRA3
        BSF     PORTD,2
        BCF     PORTD,1
        BCF     PORTD,0
        BTFSC  PORTA,2
        RETLW'7'
        BCF     PORTD,2
        BSF     PORTD,1
        BCF     PORTD,0
        BTFSC  PORTA,2
        RETLW'8'
        BCF     PORTD,2
        BCF     PORTD,1
        BSF     PORTD,0
        BTFSC  PORTA,2
        RETLW'9'
        GOTO    TUSOKU
SIRA4
        BSF     PORTD,2
        BCF     PORTD,1
        BCF     PORTD,0
        BTFSC  PORTA,3
        RETLW'*'
        BCF     PORTD,2
        BSF     PORTD,1
        BCF     PORTD,0
        BTFSC  PORTA,3
        RETLW'0'
        BCF     PORTD,2
        BCF     PORTD,1
```

```
                BSF             PORTD,0
                BTFSC  PORTA,3
                RETLW'#'
                GOTO            TUSOKU
TUSGECIK
                MOVLW           030H
                MOVWF           BEK3
                CALL            GECIK1
                DECFSZ          BEK3,F
                GOTO            $-2
                RETURN
GECIK1
                MOVLW           03H
                MOVWF           BEK1
TEK3
                MOVLW           0FFH
                MOVWF           BEK2
                DECFSZ          BEK2,F
                GOTO            $-1
                DECFSZ          BEK1,F
                GOTO            TEK3
                RETURN
                END
```

10.3. Sensörler Yardımı İle Çizgi Takip Eden Araba



Kullanılan Malzemeler;

*3 adet sensör

*1adet sensör hassasiyet ayar düğmesi

*1 adet reset düğmesi

*1adet opamp entegresi

*1adet kristal osilatör

*3 adet led

*1adet 16f877 pic
*2 adet güç transistörü
*2 adet 22 ohm 1waat direnç
*2 adet bc 256 transistör
*4 adet mercimek kondansatör
*8 adet direnç
*2 adet kondansatör
*1 adet çift motorlu oyuncak araba

Sensörler beyaz renge duyarlı olduğu için siyah zemin üzerine beyaz çizgi(sensörlerin plakasının genişliğinde) çizilir.Dönüşler ani olmazsa verim daha iyi olur.Sağa dönmek için sol motor ,Sola dönmek için sağ motor dönmeli, ileri içinde her iki motorda aktif olmalıdır.

```
LIST P=16F877
        INCLUDE   "P16F877.INC"
        __CONFIG(_WDT_OFF&_XT_OSC&_PWRTE_ON&_CP_OFF)
START
        BSF             STATUS,5
        CLRF            TRISC
        MOVLW           H'07'
        MOVWF           TRISE
        MOVLW           H'07'
        MOVWF           ADCON1
        BCF             STATUS,5
        CLRF            PORTC
        MOVLW           H'FF'
        MOVWF           PORTE
DONGU
        BTFSS           PORTE,0
        CALL            DUZ
        BTFSS           PORTE,1
        CALL            SAG
        BTFSS           PORTE,2
        CALL            SOL
        GOTO            DONGU
SAG
        MOVLW           H'02'
        MOVWF           PORTC
        RETURN
SOL
        MOVLW           H'01'
        MOVWF           PORTC
        RETURN
DUZ
        MOVLW           H'03'
        MOVWF           PORTC
        RETURN
        END
```

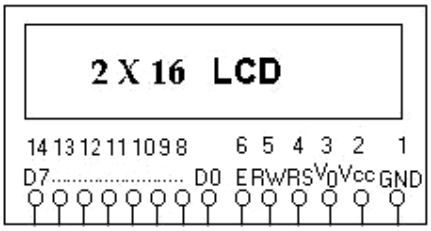
10.4. 2x16 LCD VE 4x4 Tuş Takımı ile Mesaj Yazma

Programda çıkış olarak **B Portu** çıkış olarak kullanılmaktadır. Bu port LCD'nin D0-D7 data bağlantı uçlarına bağlanır. Aynı zamanda LCD'nin E ve RS bitleri (LCD'yi aktif yapma ve Data/ komut okuma seçenekleri) PIC'in D0 ve D1 bitleri ile kontrol edilir.

Devrede kullanılan LCD 2*16'lıktır. Yani LCD 2 satır ve 16 sütundan oluşmaktadır. LCD çıkışlarından 15-16. bacaklar kullanılmamaktadır. Çünkü 15 ve 16. bacaklar LCD'nin ışığının yanması için kullanılır. LCD ışıklı olmadığı için de bu bacakları kullanmamıza gerek yoktur.

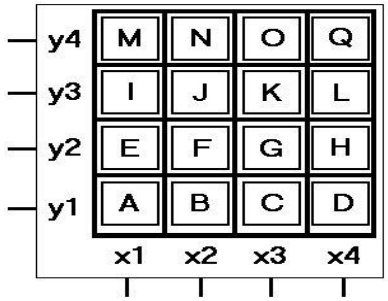
LCD'nin sadece 1. satırına bilgi yazımı gerçekleşmektedir, 2. satır kullanılmamaktadır. 16

karakter girişi gerçekleştikten sonra LCD ekranı temizlenir, yani reset atılmış olur ve kursör 1. satırın 1. sütununda yanıp sönmeye başlar. Devredeki reset butonuna basıldığında da ekran tazelenir, yani temizlenir ve kursör yine 1. satır 1. sütuna gider. Parlaklık ayarı 1., 2., 3. bitlerin uçlarının pot aracılığı ile kontrolüyle sağlanır. Aslında V0 geriliminin pot ile kontrolü ile gerçekleşir.



Devre 4*4 tuş takımıyla yani 16 harfle 2*16'lık LCD ekrana bilgi yazdırma işlemini yapıyor. Tuş takımında 16 harf olduğu için sadece bu harflerle bilgi yazımı gerçekleşmektedir (A-Q harfleri arası).

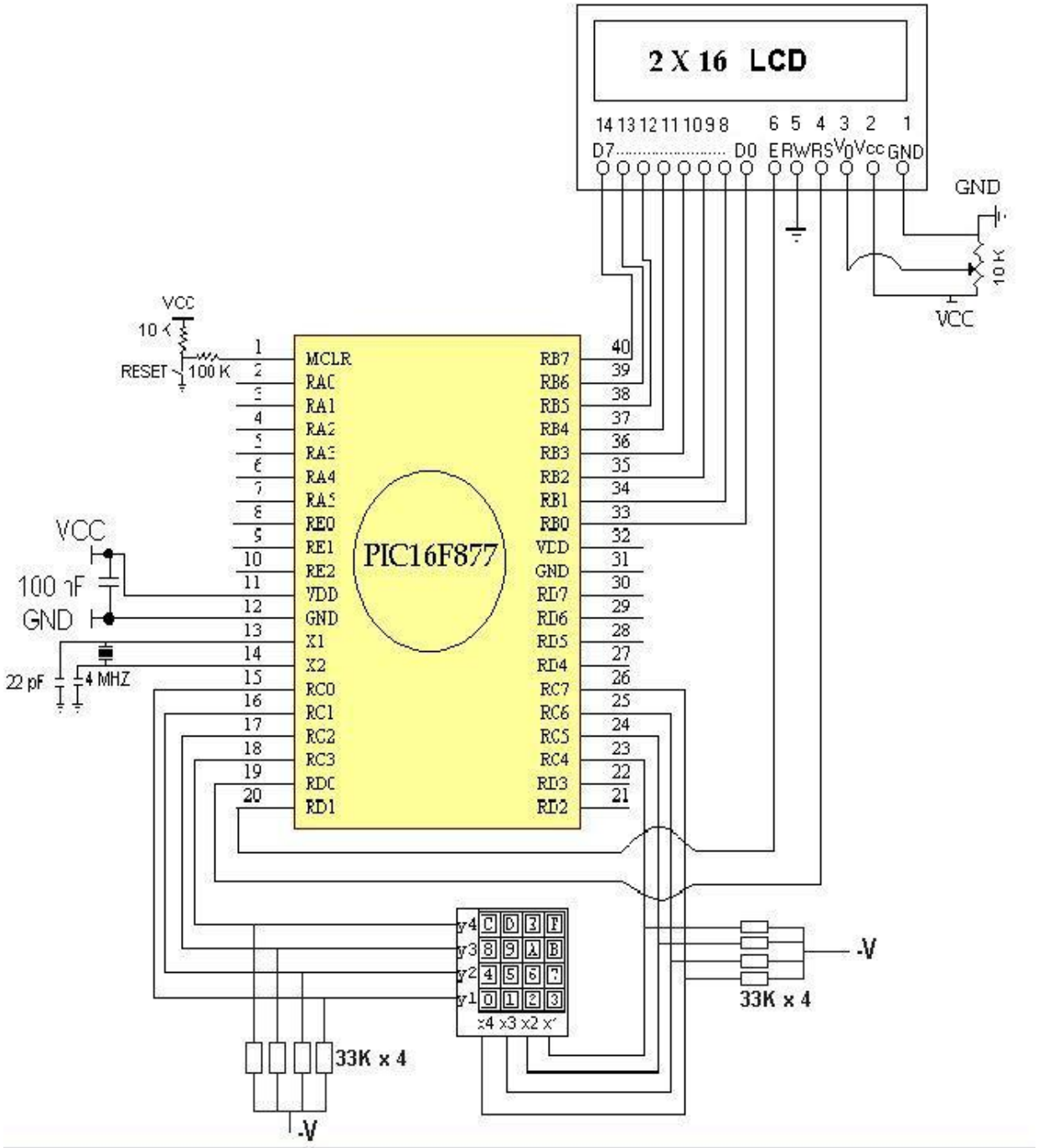
Klavyeden PIC'in C Portu ile giriş alınır. C1-C3 bitleri klavyenin Y1-Y4 satır ve C4-C7 bitleri de klavyenin X1-X4 sütun çıkışlarına bağlıdır.



PIC'imiz ve Kristal Osilatörümüz 4MHz'liktir. PIC'in 1. bacağından 10K ve 100K dirençler ve anahtar aracılığıyla çevresel kesme yani reset alınır.

Baskı devrenin şeklinden de görüleceği üzere; plaket üzerinde LCD için 16'lık ve Klavye içinde 8'lik dişi konnektör kullanılmıştır. Bu elemanlarla direk bu konnektörler aracılığı ile bağlantı gerçekleştirilir.

Aynı zamanda 8 tane 33K'lık Pull-Down dirençleri kullanılmıştır. Bunlar klavyeden bir tuşa basıldığında LCD'de kararlı görüntü alabilmeyi yani kararlı girişler alabilmeyi sağlar yani; tuş basımlarında oluşabilecek manyetik ve mekanik kararsızlıkları şaseye vererek sistemin kararlılığını artırır.




```
#INCLUDE "P16F877.INC"

SAY1            EQU     021H
SAY2            EQU     022H
RS              EQU     .0
EN              EQU     .1
COUNT_L         EQU     023H
COUNT_H         EQU     024H
CURSOR          EQU     025H
CTRL            EQU     026H
TEMP            EQU     027H

        __CONFIG        _WRT_ENABLE_OFF & _DEBUG_OFF & _BODEN_OFF & _LVP_OFF & _WDT_OFF
& _CP_OFF       & _XT_OSC & _PWRTE_OFF

        ORG             0000H
        GOTO    BASLA
        ORG             0004H
        GOTO    KESME
```

```
;*********************************************************************
;       PIC IN BASLATILMASI                                           *
;*********************************************************************

BASLA
        BSF             STATUS,RP0
        CLRF            TRISB
        MOVLW           0F0H
        MOVWF           TRISC
        CLRF            TRISD
        BCF             STATUS,RP0
        CALL            LCD_SETUP
CLEAR
        CALL            LCD_SET2
        CALL            LCD_TEMIZLE
        CALL            LCD_SET2
        CLRF            CURSOR

;*********************************************************************
;       ANA PROGRAM DONGUSU                                           *
;*********************************************************************

ENBAS
        CALL            SAT1
        ADDLW           .64
        CALL            LCD_WRITE_DIGIT
        CALL            GECIK
        CALL            SAT2
        ADDLW           .64
        CALL            LCD_WRITE_DIGIT
        CALL            GECIK
        CALL            SAT3
        ADDLW           .64
        CALL            LCD_WRITE_DIGIT
        CALL            GECIK
        CALL            SAT4
        ADDLW           .64
        CALL            LCD_WRITE_DIGIT
        CALL            GECIK
        GOTO            ENBAS

;*********************************************************************
;       KEYPAD'I KONTROL EDEN FONKSIYON                               *
;*********************************************************************

SAT1
        MOVLW           01H
        MOVWF           PORTC
        BTFSC  PORTC,4
        RETLW.1
        BTFSC  PORTC,5
        RETLW.2
        BTFSC  PORTC,6
        RETLW.3
        BTFSC  PORTC,7
        RETLW.4
SAT2
        MOVLW           02H
        MOVWF           PORTC
        BTFSC  PORTC,4
        RETLW.5
        BTFSC  PORTC,5
        RETLW.6
        BTFSC  PORTC,6
        RETLW.7
```

```
        BTFSC  PORTC,7
        RETLW.8
SAT3
        MOVLW       04H
        MOVWF       PORTC
        BTFSC  PORTC,4
        RETLW.9
        BTFSC  PORTC,5
        RETLW.10
        BTFSC  PORTC,6
        RETLW.11
        BTFSC  PORTC,7
        RETLW.12
SAT4
        MOVLW       08H
        MOVWF       PORTC
        BTFSC  PORTC,4
        RETLW.13
        BTFSC  PORTC,5
        RETLW.14
        BTFSC  PORTC,6
        RETLW.15
        BTFSC  PORTC,7
        RETLW.0
        GOTO   SAT1

;*****************************************************************
;       LCD NIN BASLATILMASI                                    *
;*****************************************************************

LCD_SETUP
        BCF         PORTD,RS
        MOVLW       B'00001111' ;KURSOR YANIP SONECEK
        MOVWF       PORTB
        CALL        LCD_ENABLE
        CALL        GECIK
        RETURN

;*****************************************************************
;       CURSOR U EN BASA ALAN FONKSIYON                         *
;*****************************************************************

LCD_SET2
        BCF         PORTD,RS
        MOVLW       B'10000000'
        MOVWF       PORTB
        CALL        LCD_ENABLE
        CALL        LCD_DELAY
        BSF         PORTD,RS
        RETURN

;*****************************************************************
;       LCD EKRANINI TEMIZLEYEN FONKSIYON                       *
;*****************************************************************

LCD_TEMIZLE
        CLRF        CURSOR
        CALL        LCD_SET2
        BSF         PORTD,RS
        MOVLW       .32
        MOVWF       PORTB
        MOVLW       .17
        MOVWF       TEMP
CLEAR
        CALL        LCD_ENABLE
```

```
        CALL    GECIK
        DECFSZ  TEMP,F
        GOTO    CLEAR
        CALL    LCD_SET2
        RETURN

;*************************************************************
;       ALT SATIRA GECME FONKSIYONU                         *
;*************************************************************

LCD_NEW_LINE
        INCF    CTRL
        BCF     PORTD,RS
        MOVLW   B'10010000'
        MOVWF   PORTB
        CALL    LCD_ENABLE
        BSF     PORTD,RS
        MOVLW   .2
        SUBWFCTRL,W
        BTFSC  STATUS,Z
        CALL    LCD_SET2
        RETURN

;*********************************************************************
;    LCD NIN DATA VEYA KOMUTU ISLEMESINI SAGLAYAN FONK    *
;*********************************************************************

LCD_ENABLE
        BCF     PORTD,EN
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        BSF     PORTD,EN
        CALL    LCD_DELAY
        RETURN
LCD_DELAY
        MOVLW   7FH
        MOVWF   COUNT_H

LCD_DELAY2
        MOVLW   0FFH
        MOVWF   COUNT_L

LCD_LOOP
        DECFSZ  COUNT_L,F
        GOTO    LCD_LOOP
        DECFSZ  COUNT_H,F
        GOTO    LCD_DELAY2
        RETURN

;*************************************************************
;       OKUNAN KAREKTERI LCD YE YAZAN FONKSIYON      *
;*************************************************************

LCD_WRITE_DIGIT
        MOVWF   PORTB
```

```
        INCF            CURSOR
        BSF             PORTD,RS
        CALL            LCD_ENABLE
        MOVLW           .16
        SUBWF CURSOR,W
        BTFSC  STATUS,Z
        CALL            LCD_TEMIZLE
        RETURN

;*****************************************************************
;GECIKME FONKSIYONU                                              *
;*****************************************************************

 GECIK                  ;193 MS BEKLEME
        MOVLW           .255
        MOVWF           SAY2
BAS1
        MOVWF           SAY1
DON2
        DECFSZ          SAY1,F
        GOTO   DON2
        DECFSZ          SAY2,F
        GOTO            BAS1
        RETURN
;*****************************************************************
;KESME          ALT PROGRAMI                                     *
;*****************************************************************
KESME
        RETFIE
        END
```

10.5. Bölüm Kaynakları

1. Öğrenci Proje Uygulamaları, 2003-2006