T.C MARMARA ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ ELEKTRİK-ELEKTRONİK BÖLÜMÜ

LİSANS ÜSTÜ BİTİRME TEZİ

TEZİN ADI

BİR KİMYASAL KAPLAMA HATTININ MİKROİŞLEMCİ İLE KONTROLU

TEZİ HAZIRLAYAN

KÂMİL ÖZÇIKMAK

TEZİ YÖNETEN

Yrd.Doç.Dr.BURHANETTİN CAN

6 EKİM 1986

Ö N S Ö Z

Yedi ana bölümden oluşan bu yüksek lisans tez çalışması ilk beş bölümü günümüzde elektronik bilim dalında oldukça geniş yer tutan mikroişlemcileri donanım ve yazılım açısından incelemiş ve bir temel mikrobilgisayar oluşturulmuştur. 6., 7. bölümlerde oluşturulan bu mikrobilgisayarla, bir kimyasal işlemler dizisi gerektiren baskılı devre hattı için kontrol prototipi ve yazılımı oluşturulmuştur. Bu kontrol sadece bir örnek olarak düşünülmüş olup, mikrobilgisayarla daha farklı kontrol donanım ve yazılımları geliştirilebilir. Bu bakımdan mikrobilgisayarla, Üniversite'deki öğrenci arkadaşlar yazılım geliştirme çalışmaları yapabilirler.

Mikroişlemciler günümüzde çok yaygın olarak, çeşitli karmaşık kontrol işlemleri ve ölçme düzenlerinde, Teleprinter, Telex, Telefax, Sayısal Haberleşme Sistemleri, Bilgisayar Sistemlerinin aralarında haberleşmeleri, iş makinalarının kontrolu, personel tipi mikrobilgisayarlar da kullanılmaktadır. Donanım çok amaçlı olarak tasarlanmasıyla, aynı donanımla çok farklı işlerin yazılım yoluyla sağlanması, mikroişlemci sistemlerinin en önemli üstünlükleridir.

Mikro işlemcilerin bilimsel çalışmalara en büyük desteği sayısal analiz yoluyla fiziksel sistemlerin benzeştirilebilmesi konusunda ki kolaylığıdır. Örnek olarak oransal, tümlevsel ve türevsel kontrol yazılımı ile bir fiziksel kontrol sisteminin davranışları oransal, tümlevsel ve türevsel kontrol katsayılarının değerine bağlı olarak incelenebilir.

./..

Mikroişlemcilerle çalışmada en büyük güçlük makina kodunun karmaşıklığı, anlamada güçlüğüdür. Ancak yazılım konusunda çalışmak makina koduna aşinalığı sağlamaktadır. Günümüzde yazılım çalışmaları genellikle yazılım geliştirme araçları (Software Devolopment System) ile yapılmaktadır. Bunun yanısıra yüksek düzeyli dillerin makina koduna derleyen sistemlerde kullanılmaktadır.

Tez çalışmasında 6800 temelli mikroişlemcilerden 6802 kullanılması makina kodunun diğer sekiz bitlik mikroişlemcilere göre daha kolay anlaşılır olmasındandır. Ayrıca 68000 onaltı bitlik mikroişlemcilerine temel oluşturmaktadır.

Ekim 1986, İstanbul

Kâmil ÖZÇIKMAK

FOREWORD

This theses is based on seven sections.

1st, 2nd, 3rd.	sections	—	The definition of microprocessig and the principles.
4th	sections	—	The hardware of the microcomputer which controls the plating line
5th	sections	—	The definition of the software
6th	section	—	The definition of the plating line
7th	section	—	The control and the modelling of the plating line

The microcomputer whic is defined in the fourth and fifth sections can be used as Eprom programmer as well. The advantage of the microprocessing systems is the ability to be adapted to different applications by changing the software based on the same hardware.

İÇİNDEKİLER

./..

./..

7. BÖLÜM - KONTROL MİKROBİLGİSAYARI, DONANIM, PROTOTİP VE YAZILIMI

7.1. Kontrol bilgisayarı donanımı, prototip

7.2. Taşıyıcı hareket diagramı

7.3. Kontrol yazılımı akış çizeneği

7.4. Kontrol yazılımı

KAYNAKÇA

./..

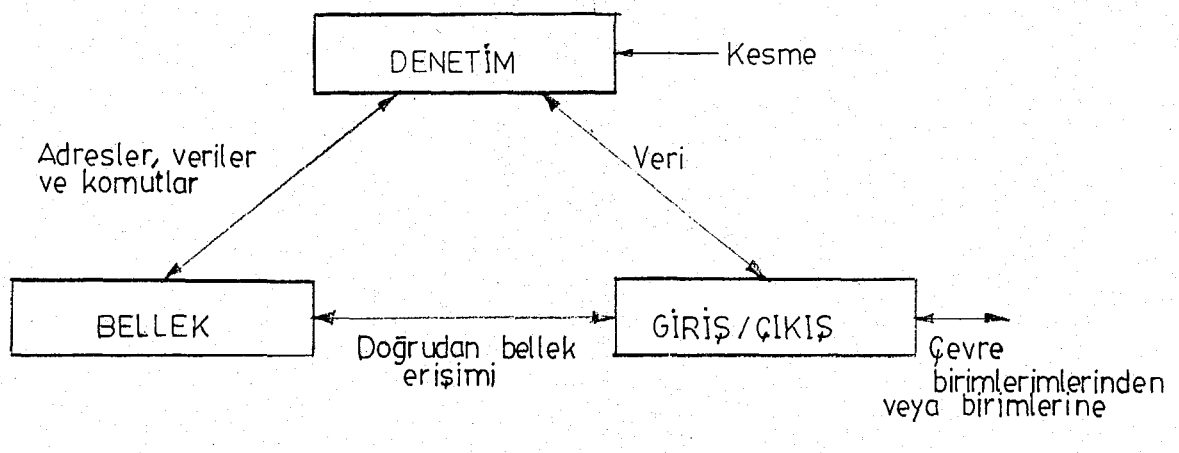
BÖLÜM 1
MİKROİŞLEMCİ YAPISI

1.1 Genel bilgisayar yapısı:

Bir bilgisayar temelde üç ana bölümden oluşmuştur: Bunlar, şekil 1.1'de de gösterildiği üzere, verilerin işlendiği ve komutların yerine getirildiği denetim bölümü, veri ve komutların saklandığı bellek bölümü ve dış dünya ile bilgisayar arasındaki iletişimi sağlayan giriş/çıkış bölümüdür. Bu bölümler arasındaki elektriksel bağlantı taşıt adı verilen yollarla sağlanır. Aşağıda her bölüm ayrı ayrı açıklanmıştır.

1.2. Bellek bölümü:

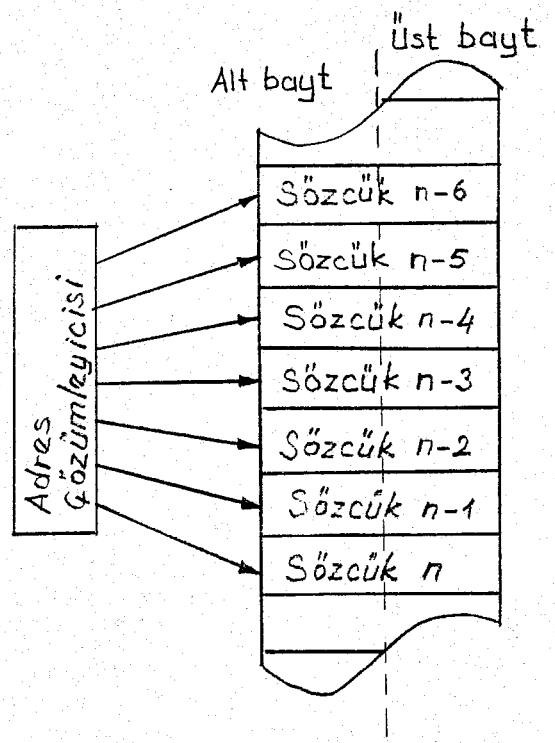
Bir bilgisayarın bellek bölümü ferrit çekirdek veya yarıiletken hücrelerden oluşturulmuş içlerinde 0 veya 1 bir şeklinde bilgi saklanan birimleri içerir. Bu birimler "baytlar" ve "sözcükler" şeklinde gruplandırılmıştır. Bellek, şekil 1.2.'de gösterildiği biçimde, ardışıl sözcükler şeklinde düzenlenmiş olup herbir bellek sözcüğünün tek bir adresi vardır.



Şekil 1.1. Bir bilgisayarın temel yapısı (1)

Şekil 1.2'de bilgisayarın sözcük uzunluğunun 2 bayt (16 bit) olduğu varsayılmıştır. Bellek bölümünün bir parçası olan adres çözümleyici, denetim biriminin gösterdiği adresi alır ve bu adrese sahip bellek konumunu seçerek veri taşıtına bağlar. Adresin gösterdiği bellek konumunun doğru olarak seçilmesi ve buradaki bilginin veri taşıtına aktarılması (okuma) veya veri

taşıtındaki bilginin buraya işlenmesi (yazma) belirli bir zaman gerektirir. Bu zamana "bellek erişim zamanı" adı verilir. Bilgisayar bellekleri genellikle "rasgele erişim" türünden olup erişim zamanları 3 nanosaniye'den birkaç mikrosaniyelere kadar değişir. Burada rastgele erişim terimi, belleğin herhangi bir konumuna, oraya yazmak veya oradan okumak için gerekli sürenin herhangi başka bir konum için gereken ile aynı olduğu anlamını verir.



Şekil 1.2. Sözcük uzunluğu 16 bit olan bir bilgisayar için bellek organizasyonu, [Microprocessor data book, Motorola]

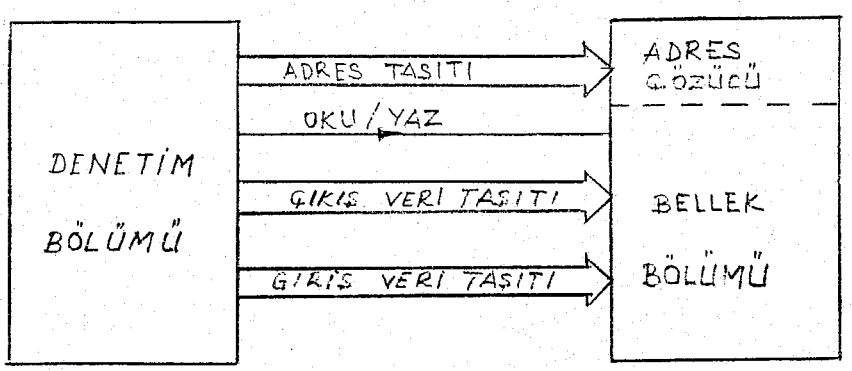
Yukarıda açıklanan bellek bölümü "anabellek" veya "birincil bellek" olarak da adlandırılır. Bilgisayara çevresel birim olarak bağlanan ve çok büyük boyutlu bilgi saklıyabilen ikinci bir bellek bölümü daha olabilir. Böyle bir bölüme "ikincil bellek" adı verilir ve kalıcı manyetik özellikli ortamda yapılır. İkincil bellek birimleri ardışık erişimli olur, yani aranan bellek konumuna erişmek için peşpeşe diğer konumları gezerek ilerleme gerekir. Dolayısıyla da bir bellek konumuna erişim konumdan konuma değişir. Ardışık erişim, sıralı erişim veya doğrudan erişim türünden olabilir. Sıralı erişimde adresleme yoktur. Bilgi depolama ortamı olarak ses kasetlerine benzer manyetik şerit kullanılır ve bellek konumları baştan başlana-

rak ard arda okunur. Doğrudan erişime örnek olarak ise disk bellekleri verebiliriz. Bu tür bellek birimlerinde adresleme yapılmakla birlikte verilen bir adrese ulaşabilmek için geçerli süre, okuyucu kafanın o adrese ne kadar yakın veya uzak olduğuna göre değişir.

Belleğin belirli bir konumdan yazma veya okuma şu şekilde gerçekleştirilir:

1. Denetim bölümü, bellek bölümüne belirli bir adres gönderir.
2. Denetim bölümü, verinin transfer yönünü göstermek için bellek bölümüne bir OKU/YAZ (READ/WRITE) işareti gönderir.
3. Denetim bölümü transferin gerçekleşmesi için gerekli bir süre bekler. Bu süre okuma işleminde transferden önce, yazma işleminde ise transferden sonra yer alır.

Denetim ve bellek bölümleri birbirlerine çeşitli taşıtlarla bağlanırlar. Adres taşıtı, denetim bölümünü kullanmak istediği bellek konumunun adresini taşır. OKU/YAZ işareti verinin transfer yönünü (denetim bölümünden bellek bölümüne veya bellek bölümünden denetim bölümüne) gösterir. Veri taşıtı veriyi bölümler arasında taşımakta kullanılır. Şekil 1.3'te okuma ve yazma için iki ayrı taşıt gösterilmiş olmasına rağmen gerçekte bu taşıtlar tek bir taşıt olabilir ve hatta adres için de aynı taşıt kullanılabilir. Böyle bir durumda "taşıt zaman-paylaşmalı" ("time-shared", "time multiplexed") bir şekilde kullanılır.



Şekil 1.3. Denetim ve bellek bölümü arasındaki bağıntılar
[mikroprocessor data book, Motorola]

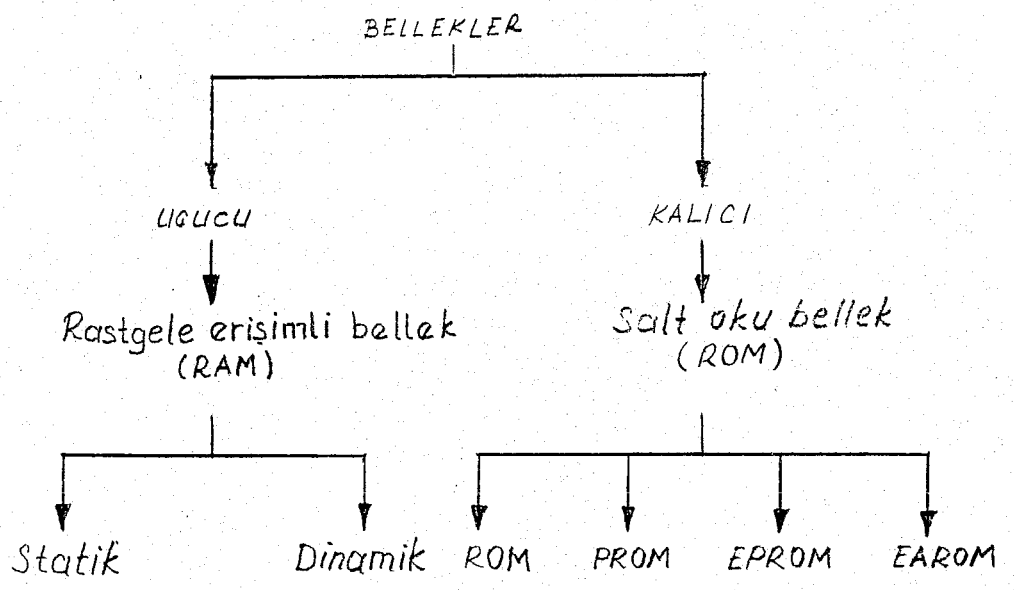
Bilgisayarın önemli bir özelliği gerek verilerin gerekse komutların aynı her bir gösterimle (1'ler ve 0'lar dizisi ile) aynı bellek bölümü içinde yer almasıdır. Bu tür bir makinaya Von Neumann makinası adı verilir. Bellekteki herhangi bir bayt komut veya veri olabileceğine göre bilgisayarın bunları nasıl ayırdığı sorusu akla gelebilir. Bilgisayarın denetim bölümü, bellek bölümünde okunan bilginin ne zaman bir veri ne zaman bir komut olduğunu bilir. Yazılımcı bir hata yaparsa komut bilgi veya bilgi komut olarak yorumlanabilir ve sonuçta bilgisayar yanlış bir işlem içine girer.

1.2.1. Bellek türleri:

Altbölüm 1.1'de bir bilgisayarın bellek bölümünün ferrit çekirdek veya yarıiletken hücrelerden oluşturulduğuna değinilmişti. Bu altbölümde bellek çeşitleri açıklanacaktır.

Ferrit çekirdek, ferromagnetik seramik malzemeden yapılmış minyatür bir halka olup içinden geçen akım yönüne göre bir yönde ya da diğer yönde mıknatıslanır. Halkanın bu kalıcı mıknatıslanma özelliği ikili bilgi saklamak için yakın zamanlara kadar yaygın olarak kullanılıyordu. Günümüzde artık yerini, özellikle birincil bellek ortamı için, yarıiletken malzemeden yapılmış bellek hücrelerine bırakmıştır.

Yarı iletken bellek elemanlarının yaygın olarak kullanılan türleri Şekil 1.4.'te gösterilen şekilde sınıflandırılabilir. Aşağıda herbir bellek türü ayrı ayrı açıklanmıştır.



Şekil 1.4. Yarı iletken bellek türleri

Kalıcı (non-volatile) bellek: Bu tür bir bellekte bilgi kalıcıdır, besleme gerilimi kesildiğinde kaybolmaz.

Uçucu (volatile) bellek: Bu tür bellekte besleme gerilimi kesildiği zaman saklı bilgi kaybolur.

Rasgele erişimli bellek (Random Access Memory-RAM): Bu tür bellekler genellikle bipolar (iki-kutuplu) veya MOS (metal-oxide-semiconductor) üretim teknikleri kullanılarak imal edilirler ve hem okunabilme hem de yazılabilme özelliklerinden dolayı yaz-oku bellek olarak da adlandırılırlar. Statik ve dinamik olmak üzere iki genel gruba ayrılırlar. Dinamik RAM belleklerde bilgi kapasitif bellek hücrelerinde saklandığından 2-3 milisaniye içerisinde bilgi kaybolur. Bilginin devamlı olarak hücrede kalabilmesi için bellek hücresinin periyodik bir şekilde (genellikle her 1-2 milisaniyede bir) tazelenmesi gerekir. Bu işlem için özel dinamik bellek tazeleme devreleri geliştirilmiştir. Statik bellekler tazeleme işlemi gerektirmezler fakat daha pahalı olup daha fazla güç harcarlar. Yapımları daha karmaşık olduğu için tek bir yonga üzerine yerleştirilebilecek statik RAM bellek gözü, dinamik RAM bellek gözüne göre daha az olur.

Salt oku bellek (Read Only Memory - ROM) : Bu tür bellek yalnız okunabilir fakat içine bilgi yazılmaz. Bilgisayarlardan en yaygın olarak kullanılan bellek türüdür. Yine genellikle bipolar veya MOS teknoloji kullanılarak üretilirler ve üretim esnasında içlerine bilgi, kullanıcının arzusuna uygun bir şekilde "yakılarak" depolanır. Bu bilgi bir daha değiştirilemez. En yaygın olarak kullanılan bellek türü olup erişim zamanları ve organizasyonları çok çeşitlilik gösterir. Yaygın kullanım alanlarına örnek olarak değiştirilmemesi gereken yazılım depolanması, sinüs kosinüs, logaritma gibi fonksiyon tabloları, çarpma, bölme tabloları, yazıcılar ve CRT ekranlar için 5x7 "dot-matrix" pattern üreticileri gösterilebilir.

Programlanabilir salt oku bellek (Programmable Read Only Memory-PROM): ROM belleğin bir türü olup kullanıcı tarafından, onun isteğine uygun bir şekilde, özel gereçlerle yalnız bir defa için programlanabilir. ROM bellekten daha pahalı fakat RAM bellekten daha ucuz olup daha çok üretim yoğunluğu fazla olmayan amaçlar için kullanılır.

Silinebilen programlanabilir salt oku bellek (Erasable PROM-EPROM): Özel bir PROM türü olup içeriği kullanıcı tarafından silinebilir ve yeniden programlanabilir. Bu amaçla yongaanın muhafatası üzerinde bir pencere açılmıştır. Mor-ötesi ışık kullanarak bellek içerisindeki bilgi silinir. Yeniden programlama

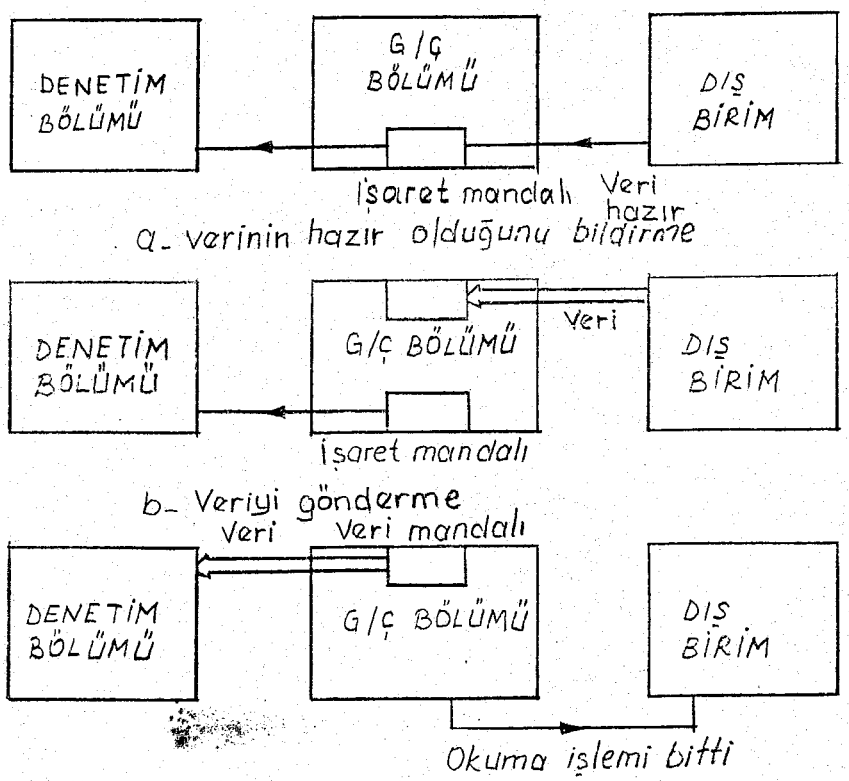
için özel bir gereç (EPROM Programmer) gerekir. Özellikle ürün geliştirme sırasında yazılım geliştirme ve deneme amaçları ile kullanılır.

Elektriksel yolla içeriği değiştirilebilen salt oku bellek (Electrically Alterable ROM-EAROM): ROM gibi kalıcı bir bellek ortamı olup içeriği elektriksel olarak değiştirilebilir.

1.3. Giriş/Çıkış (G/Ç) Bölümü:

Giriş/Çıkış (Input/Output - I/O) bölümü bilgisayar çevre birimleri ve dış devreler arasında veri transferini gerçekleştirir. Bilgisayar bu birimler arasında zamanlama ve hız açısından farklılıklar olabileceği gibi bilgisayarın gerektirdiği gerilim ve akım düzeyi ile dış biriminin verebileceği gerilim ve akım düzeyleri de farklı olabilir. G/Ç bölümünün bu farklılıkları ortadan kaldırması ve dıştan gelen bilgiyi bilgisayara uygun bir biçime sokması gerekir. Kesikli bir şekilde gelen bilgiler için kullanılan kesme işaretlerinin ve denetim işaretlerinin işlenmesi yine G/Ç bölümü içinde yapılır.

Bir dış birimden bilgisayara bilgi aktarımı genellikle şekil 1.5.'te gösterilen biçimde gerçekleştirilir.



Şekil 1.5. Bir dış birimden bilgisayara bilgi aktarımı

a. Dış birim, G/Ç bölümü aracılığı ile denetim bölümüne yeni bir bilginin hazır olduğunu bildirir. G/Ç bölümü bu işareti bilgisayar için uygun bir biçime sokar ve denetim bölümü kabul edinceye kadar tutar.

b. Dış birim bilgiyi G/Ç bölümüne gönderir. Bu bilgi denetim bölümü tarafından okununcaya kadar G/Ç bölümünde tutulur.

c. Denetim bölümü bilgiyi okur. Okuma işlemi tamamlanınca "veri var" işareti otomatik olarak sıfırlanır ve dış birimine okuma işleminin bittiğini gösteren bir "teyid işareti" (acknowledge) gönderilir.

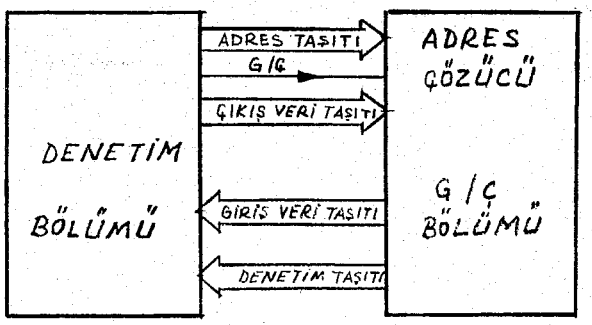
Burada şunu belirtmek gerekirki G/Ç bölümüne birçok dış devreden bilgi gelebilir. Dolayısıyla da burada bütün bu bilgilerin okununcaya kadar saklanabileceği bölümler (kapılar) ve ayrıca, bir de, bilgisayarın okumak istediği kapıyı (portu) seçecek bir çözümleyici bulunmalıdır.

Bir dış birime bilgi aktarımı, oradan bilgi girişine benzer bir şekilde gerçekleştirilir. Dış birim bilgisayara bilgi okumaya hazır olduğunu bildirdiği zaman bilgisayar bilgiyi, dış birime bilginin hazır olduğunu bildiren bir işaretle (strobe) birlikte yollar. G/Ç bölümü bilgiyi ve denetim işaretlerini formatlar ve dış birim alıncaya kadar içerisinde tutar.

G/Ç bölümü bazı durumlarda denetim bölümünün bazı görevlerini de üstlenebilir. Bunlar, bilginin seri biçimden (bir anda bir bit) paralel biçime (bir anda birden fazla bit) sokulması, bilgi başına ve sonuna bilginin başlangıcının ve sonunu belirten özel bit dizilerinin eklenmesi, daha sonra göreceğiniz ve hata sezilmesine yarayan paritenin kontrolu gibi görevler olabilir. G/Ç bölümü bu görevleri, denetim bölümünün yazılımla gerçekleştirebileceğinden çok daha kısa bir sürede donanımla gerçekleştirebilir. G/Ç bölümü programlanabilen bir özellik de taşıyabilir ve hatta bazı bilgi işlemelerini yapabilmesi için bir mikroişlemci de içerebilir.

Şekil 1.6.'da denetim ve G/Ç bölümleri arasındaki bağlantılar gösterilmiştir. Adres taşıtı denetim bölümünün kullanmak istediği giriş veya çıkış kapısının adresini taşır. Giriş/Çıkış işareti bilgi transferinin hangi yönde gerçekleştirilmesi gerektiğini bildirir. Veri taşıtları bölümler arasında bilgi taşır. Denetim taşıtında ise verinin hazır olduğunu, veri transferinin tamamlandığını bildiren işaretler gibi denetim işaretleri içinde yer alır. Bellek ve denetim bölümleri arasındaki taşıtlarda olduğu gibi bu taşıtların bazıları da fiziksel olarak aynı olabilir ve değişik amaçlar için zaman bölüşmeli (time-shared) bir

şekilde kullanılabilirler. Buna ek olarak bazı bilgisayarlarda bellek ve G/Ç bölümleri aynı taşıtlarla denetim bölümüne bağlı olabilir. Bu durumda denetim taşıtının bir hattındaki işaretin durumu hangi bölümün taşıta bağlanacağına belirtir. Bazı bilgisayarlarda ise bellek ve G/Ç bölümleri tek bir bölüm şeklinde birleştirilmiştir. Örneğin Motorola ailesindeki mikrobilgisayarlarda bir G/Ç kapısı denetim bölümüne aynen bir bellek gözü gibi gözükür. (G/Ç "memory-mapped" durumundadır).

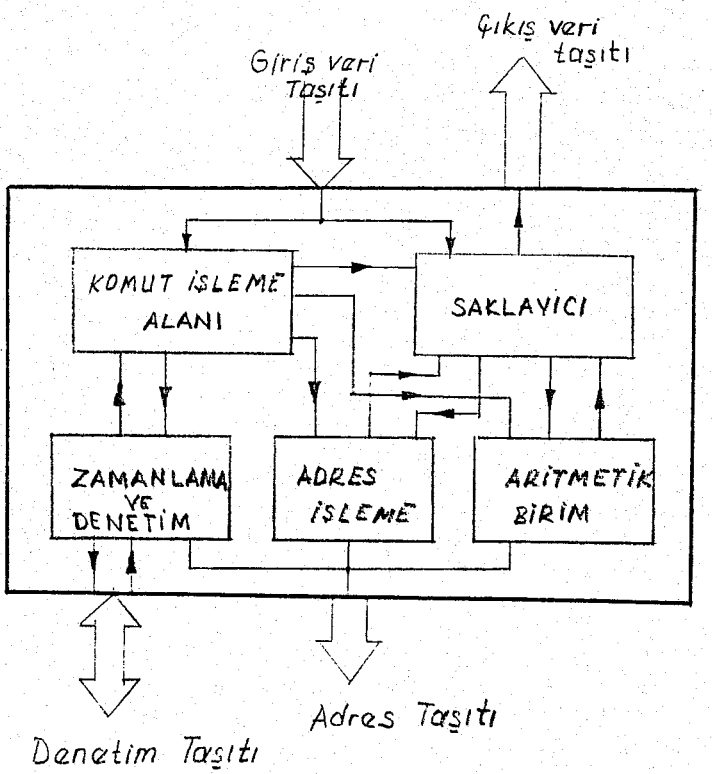


Şekil 1.6. Denetim ve G/Ç bölümleri arasındaki bağıntılar
[Motorola, Microprocessor data book]

Modern bilgisayarlarda bellek ve G/Ç bölümleri arasında doğrudan bir bağlantı vardır. Böylece denetim bölümü araya girmeden bellekten bir G/Ç kapısına veya bir G/Ç kapısından belleğe doğrudan bilgi aktarımı çok hızlı (denetim bölümünün aracı olduğu durumundan 10-20 kat daha hızlı)bir şekilde gerçekleştirilebilir. Bu tür bir işleme Doğrudan Bellek Erişimi (Direct Memory Access - DMA) adı verilir.

1.4. Denetim Bölümü:

Bilgisayarın en önemli birimi olan bu bölüm ana görevi bilgi işlemektir. Dolayısıyla da Merkezi İşlem Birimi (Central Processing Unit -CPU) olarak adlandırılır. MİB bellekten komutları alır, çözümler, yerine getirir, zamanlama ve denetim işaretlerini üretir, bellek ve G/Ç bölümlerinden veya bölümlerine veri transfer eder, veri üzerinde aritmetik ve mantık işlemleri yapar ve dıştan gelen işaretleri (kesme gibi) tanır ve gereğini yapar. Şekil 1.7'de tipik bir MİB'nin öbek şeması gösterilmiştir.



Şekil 1.7. Bilgisayarın denetim bölümünün (Merkez İşlem Biriminin) yapısı. [Microprocessor data book, Motorola]

Bir komutun yerine getirilmesi sırasında MİB'nin yaptığı işler aşağıda sıralanmıştır:

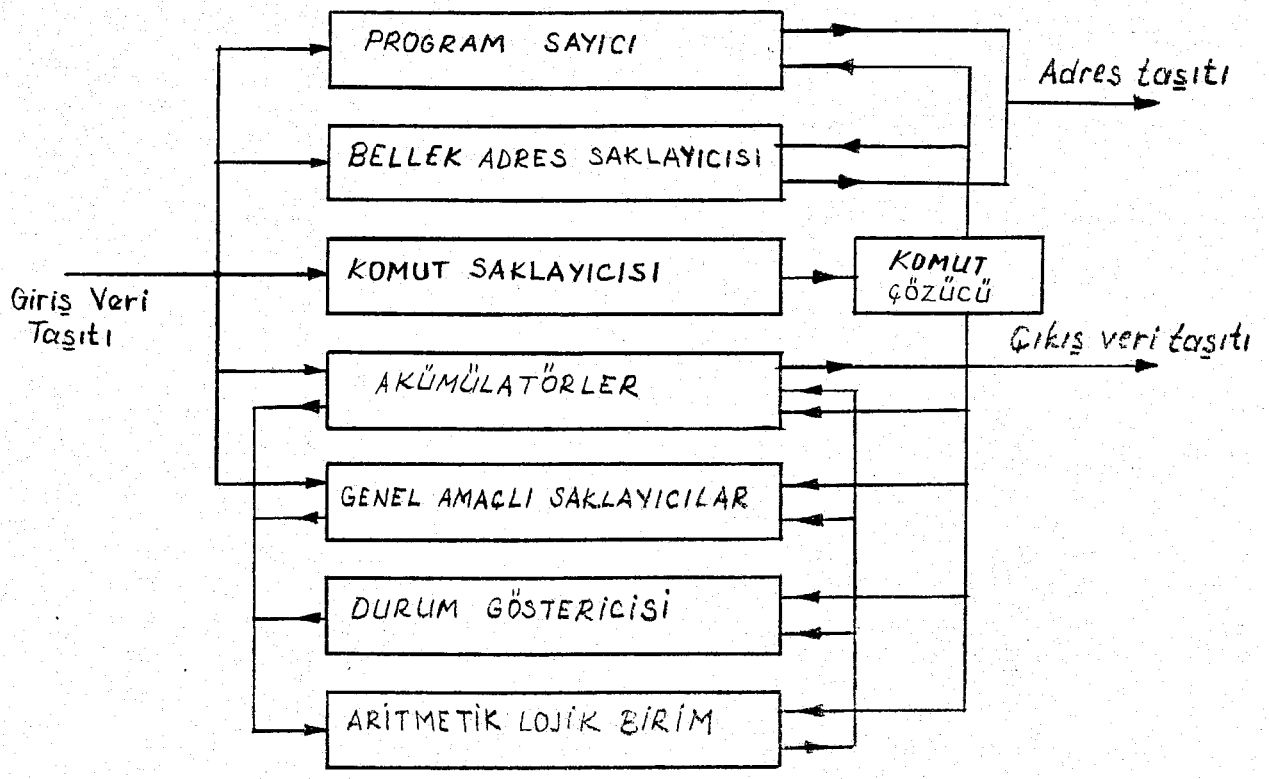
1. Komutun adresini adres taşıtına çıkarır.
2. Komutu veri taşıtından alır ve kodunu çözer.
3. Komut kodunun belirttiği işlemi yerine getirir. Bu bir aritmetik veya mantık işlemi, bir veri transferi veya idari bir işlem olabilir.
4. Komutun gerektirdiği adresleri ve veriyi içeri alır. Bunlar bellekte veya saklayıcılarda olabilir.
5. Kesme işareti gibi denetim işaretlerine bakar ve gereğini yerine getirir.
6. Bellek ve G/Ç bölümlerinin kullanımı için durum, denetim ve zamanlama işaretlerini üretir.

Yukarıda belirtilen işlemlerden de anlaşılabileceği gibi MİB bilgisayarın kalbi durumundadır. Bilgisayarın bütün birimleri MİB'in denetimi altında çalışır.

1.5 Saklayıcılar (Registers)

Saklayıcılar MİB'in ufak birer bilgi depolama birimleridir ve diğer bellek birimleri gibi ikili (binary) hücrelerden (flip-floplardan) oluşturulmuşlardır. Bazıları yazılım kontrolu altındadır ve MİB'ne herbir bilgi alış-verişi için bellek bölümüne başvurmama olanağını sağlar. Böylece işlem süresi çok kısaltılmış olur. Diğer bazıları ise denetim için gerekli bilgileri saklar.

Saklayıcılar birbirlerine, MİB'in diğer bölümlerine ve dış taşıtlara iç taşıtlarla bağlıdırlar ve iki katı uzunlukta saklayıcılar ve iç taşıtlar da olabilir.



Şekil 1.8. Saklayıcılar ve ara bağlantıları [Mikro işlemciler, Ömer Cerid]

MİB içerisinde ne kadar çok saklayıcı olursa bilgisayarın işlemleri o kadar kolaylıkla yapabileceği açıktır. Büyük bilgisayarlarda düzinelerce saklayıcı olabilir. Mikrobilgisayarlarda ise yonga büyüklüğü saklayıcı sayısı için bir üst sınır getirir. Şekil 1.8'de tipik bir saklayıcılar grubu gösterilmiştir.

Saklayıcıların değişik amaçları olabilir ve hatta, bazı bilgisayarlarda, bazılarının görevi kullanıcı tarafından yazılımla belirlenebilir. Bununla birlikte bilgisayarların çoğunda saklayıcıların görevi önceden belirlenmiştir. Hemen hemen her bilgisayarda rastlanan saklayıcı türleri aşağıda sıralanmıştır.

- Program sayıcısı (program counter-PC)
- Komut saklayıcısı (Instruction register-IR)
- Bellek adres saklayıcısı (Memory address register-MAR)
- Bellek veri saklayıcısı (Memory data register-MDR)
- Akümülatörler (Accumulators)
- İndis saklayıcıları (Index registers)
- Durum saklayıcısı (Condition code register-CCR) veya status register-SR)
- Yığın göstericisi (Stack pointer-SP)
- Genel amaçlı yazmaçlar (General purpose registers)

Program sayıcısı: Bir sonra işlenecek komutun bulunduğu bellek konumunun adresini içerir. Komut çevrimi (instruction cycle) MİB'nin program sayıcısının içeriğini (program sayısını) adres taşıtına koyması ile başlar. Böylece komutun ilk sözcüğü bellekten MİB'ne alır. Aynı zamanda program sayıcı bir arttırılır. Dolayısıyla da, eğer ATLA (JUMP) veya DALLAN (BRANCH) gibi bir komut program sayıcısının içeriğini değiştirmezse, komutlar bellekten ardışık bir şekilde MİB'ne alır.

Komut sayıcısı: Komut çözümleninceye kadar burada tutulur. Genellikle yazılımcı tarafından ulaşılamayan bir saklayıcıdır.

Bellek adres saklayıcısı: Ulaşılmakta olan verinin adresini içerir.

Bellek veri saklayıcısı: Adreslenmiş olan bellek konumuna yazılmakta veya o konumdan okunmakta olan veriyi içerir.

Akümülatörler: Aritmetik veya mantık işlemleri sırasında kullanılan geçici bilgi saklayıcılarıdır. Bazı bilgisayarlarda bir, bazılarında ise birden fazla akümülatör bulunur. Birden fazla akümülatör hesap işlemlerini önemli derecede kolaylaştırır.

İndis saklayıcıları: Adresleme işlemi için kullanılırlar. Kullanılış şekilleri daha ileride ayrıntılı olarak açıklanacaktır.

Yığın göstericisi: Bir bilgisayarda yığın önemli bilgilerin geçici olarak sakladığı bir bellek bölümüdür. Yığın göstericisi ise yığının üstünü gösterir. Yığın ve yığın göstericisi kavramları daha ileride ayrıntılı olarak açıklanacaktır.

Durum saklayıcısı: MİB'nin içerisindeki "durumu" gösteren ve bayrak adı verilen flip-floplar grubudur. Bilgisayarın karar verme mekanizmasının temeli bu bayraklardır ve sayıları bilgisayardan bilgisayara değişebilir. Yaygın olarak rastlanan bayrak türleri aşağıda açıklanmıştır.

ELDE (CARRY): MİB'nin gerçekleştirdiği son işlem bir elde sonucu doğurmuşsa bu bayrak kalkar, yani flip-flop 1 durumuna geçer. Özellikle "Multiple precision" aritmetik işlemlerinde kullanılır. Toplama işleminde elde, çıkarma işleminde ise bir ödünç durumunun olup olmadığını gösterir.

SIFIR(ZERO): Son işlemin sonucu sıfırsa 1 durumuna geçer.

TAŞMA (OVERFLOW): Bir aritmetik işlemin sonucunun sözcük kapasitesini geçip geçmediğini gösterir. Aşağıda belirtilen durumlarda bir TAŞMA var demektir ve durum saklayıcısındaki ilgili bit 1 durumuna geçer.

- Pozitif iki sayının toplamı negatif sonuç verirse
- Negatif iki sayının toplamı pozitif sonuç verirse
- Pozitif bir sayıdan pozitif bir sayı çıkarıldığında sonuç negatif çıkarsa

EKSİ (NEGATIVE): İşlem sonucu negatifse 1 durumuna geçer, sıfır veya pozitifse sıfırlanır. İŞARET (SIGN) bayrağı olarak da bilinir.

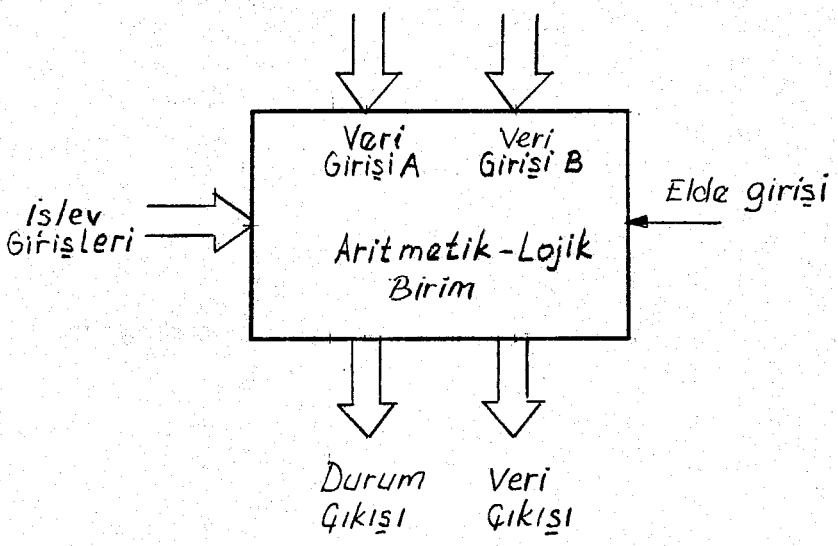
YARIM ELDE (HALF CARRY): Son işlemde alt (düşük anlamlı) dörtlü bir elde sonucu doğurursa 1 olur. BCD aritmetikte kontrolu gerekli bir bayraktır.

KESME MASKE (INTERRUPT MASK): Bu flip-flop 1 durumunda ise MİB'ne dıştan gelen bir kesme işlemine izin veriliyor anlamını verir. Bu durumda altında 1 veya 0 yapılması ile bilgisayar kesme istemlerine duyarlı veya duyarsız bir duruma getirilebilir. Bazı bilgisayarlarda bu bayrak KESMEYE İZİN (INTERRUPT ENABLE) bayrağı olarak da adlandırılır.

1.6. Aritmetik Birim:

MIB'nin bu bölümünde gerekli aritmetik ve lojik işlemleri gerçekleştirilir. Bolayısıyla da aritmetik-lojik birimi (arithmetic-logic-unit-ALU) olarak adlandırılmıştır. Şekil 1.9'da tipik bir ALU'nun yapısı gösterilmiştir. Giriş olarak bilgisayarın temel sözcük uzunluğunda iki veri girişi, ALU'nun yapacağı işlemi belirten bir işlev girişi ve ayrıca bir de ELDE girişi vardır. Çıkış olarak ise işlem sonucunu taşıyan bir veri çıkışı ve

bir de durum çıkışı vardır. Durum çıkışı durum saklayıcısana bağlı olup buradaki bayrakları yapılan işlemin sonucuna göre 1 veya 0 durumuna getirir.



Şekil 1.9. Aritmetik-lojik biriminin yapısı

İşlev girişindeki veri, ALU'nun gerçekleştireceği işlevi belirler. Bunlar

- Toplama
- Çıkarma
- Lojik VE
- Lojik VEYA
- Lojik VAR VEYA (Exclusive OR)
- Tümlev alma (Complement)
- Kaydırma (Shift)
- Arttırma (Increment)
- Eksilme (Decrement)
- Sıfırlama (Clear)

gibi işlemler olabilir. Çarpma, bölme, logaritma, v.s. gibi diğer aritmetik işlemler özel devreler ile gerçekleştirilebilir. Böylece hızlı bir çalışma elde edilir, fakat, bu tür ALU'lar standart ALU'lara kıyasla çok daha pahalıdırlar.

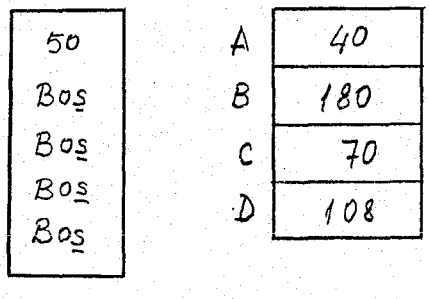
1.7. Yığın:

Yığın daha önce de belirtildiği gibi verilerin veya saklayıcı içeriklerinin geçici olarak sakladığı bir bellek bölümüdür. Buraya hem yazılması hem de okunması gerekeceğine göre belleğin RAM türü olması gerekir. Kullanım amaçları arasında kesme denetimi, altyordam bağlantıları, verilerin program denetimi altında

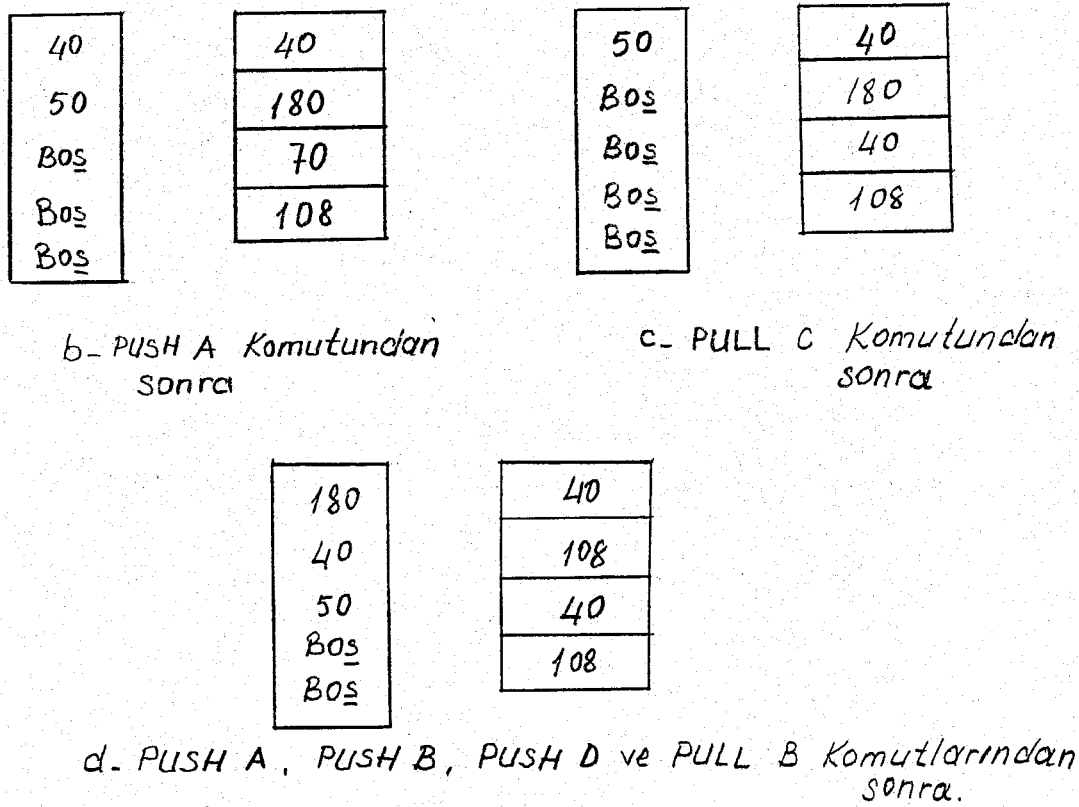
geçici olarak saklanması sayılabilir. Örneğin, bilgisayarın dışardan gelen bir istek (kesme) üzerine yürümekte olduğu programdan çıkıp başka bir programa atlaması gerekirse o andaki bütün gerekli bilgiler (saklayıcı içerikleri) yığında saklamır. Kesme üzerine atlanan program sayısına tekrar geri dönülür ve bütün saklayıcılar kesmeden önceki duruma getirilerek ana programa devam edilir.

Yığında bilgiler son-giren ilk-çıkar (last-in first-out-LIFO) bir şekilde saklanır, yani bilgiler yığının sadece üstüne konulabilir veya üstünden alınabilir. Yığına bir bilgi itilmesi PUSH, yığından bilgi alınması ise POP veya PULL olarak adlandırılan komutlarla gerçekleştirilir. Şekil 1.10'da 5 sözcük kapasiteli bir yığına A, B, C, ve D olarak adlandırılan dört saklayıcının içeriklerinin nasıl itilebileceğine ve alınabileceğine örnekler verilmiştir. Yığında başlangıçta bir sözcüğün 50 olduğunu, diğerlerinin boş olduğunu varsayarak (şekil 1.10.a) PUSH A komutu A saklayıcısının içeriğinin yığına şekil 1.10.b'de gösterilen şekilde girmesini sağlayacaktır. Bir PULL C (veya POP C) komutu ise yığını en üstündeki 50 bilgisinin C saklayıcısına girmesi sonucunu doğurur (Şekil 1.10.c). PUSH A, PUSH B, PUSH D ve PULL B gibi komutlar dizisinin sonunda ise durum Şekil 1.10.d'da gösterdiği gibi olacaktır.

Gerçekte yığına bilgi itilmesi veya alınması sırasında yığındaki bilgiler Şekil 1.10'da gösterilen şekilde bir aşağıya kaymazlar veya bir yukarıya çıkmazlar. Tek değişiklik yığın göstericisinin gösterdiği bellek konumda olur. PUSH komutundan sonra yığın göstericisi otomatik olarak bir azalır. Böylece yığın göstericisi boş bir adresi gösterir. PULL işleminde ise yığın göstericisi önce otomatik olarak bir artar ve durumda gösterdiği konumdaki bilgi istenilen yere alınır.



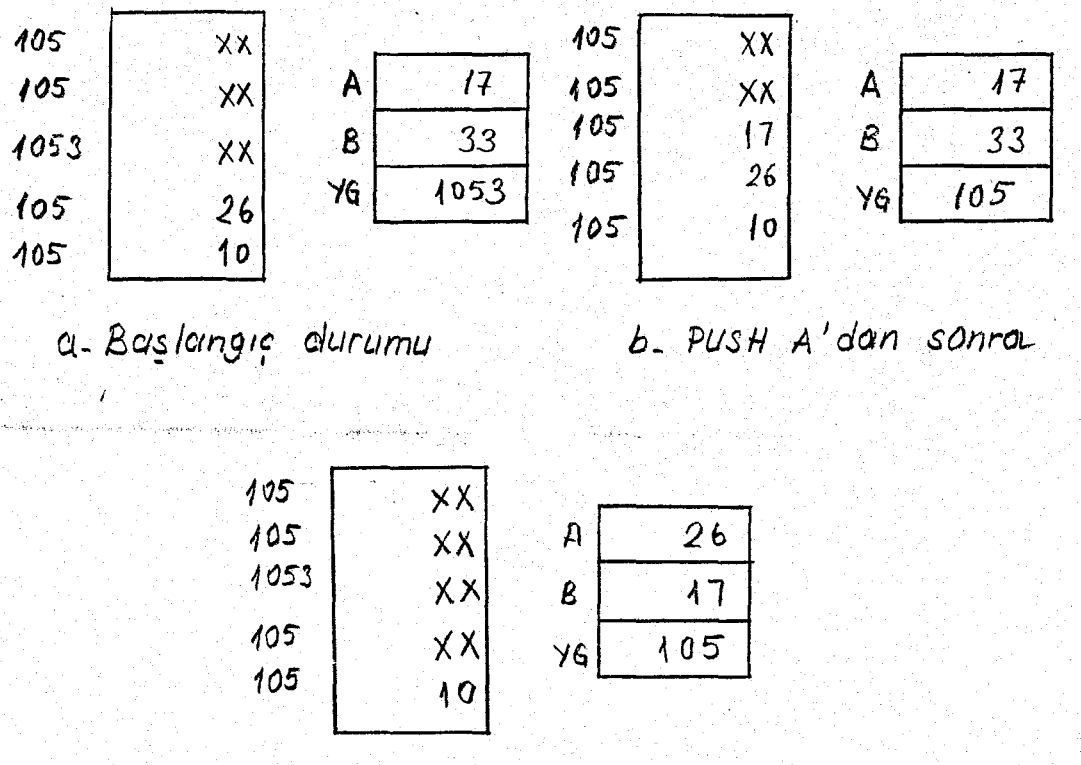
a- Başlangıç



Şekil 1.10. Yığın kullanımına bir örnek

Yığın göstericisinin gösterdiği konum yine bir boş konum olmuş olur. Burada "boş konum" o bellek gözünde hiçbir bilgi olmadığı anlamına alınmalıdır. Bir bellek gözünde flip-floplar daima 1 veya 0 durumda olmak zorundadır. Boş konum deyimi oradaki bilginin "gereksiz" (redudant) olduğu anlamına kullanılmıştır. Şekil 1.11'de örnek verilmiş olup buradaki XX'ler gereksiz bilgileri göstermektedir.

Yığın programcıya birçok kolaylıklar sağlar. Fakat kullanımında dikkatli olmak gerekir. Yığın kullanımında yapılan hataların bulunması çok zor olabilir. Tipik olarak yapılan hatalar arasında yığından bilgilerin yanlış bir sırada alınması, yığına istenmeyen bilgilerin fazladan sokulması veya alınması, yığına çok fazla bilgi sokmaya çalışarak yığının taşmasına neden olunması (yığın göstericisinin yığın olarak ayrılmış bellek bölümünün altını veya üstünü gösterir duruma gelmesi ki buralarda hiçbir bellek olmayabilir) sayılabilir.



Şekil 1. 11. Yığın işlemleri sırasında yığının ve yığının göstericisinin durumuna bir örnek

1.8. Mikroişlemci mimarisinin özellikleri:

Mikroişlemcilerin mimarisi büyük bilgisayarlara göre birçok açıdan farklılıklar gösterir. Saklayıcıları ele alacak olursak aşağıda sıralanan nedenler mikroişlemci saklayıcılarının daha değişik bir mimariye sahip olmalarını gerektirir.

- Sınırlı yonga büyüklüğü. Bir tek-yonga mikroişlemcide saklayıcı ve taşıtlar dar ve sınırlı sayıda olmak zorundadır.
- Program için salt oku bellek kullanımı. Mikroişlemciler dolayısıyla programda adres veya veri saklayamazlar.
- Sınırlı yaz-oku bellek
- Kısa sözcük uzunluğu. Bir bellek adresi birden fazla sözcük gerektirebilir.
- Kesmeli çalışma. Saklayıcılar, kesmelerin kolayca sezilip hizmet verilebilmelerine olanak vermelidirler.

Yukarıda sıralanan unsurların bazıları tasarımcıları bir ikilem içine sokacak niteliktedir. Yonga büyüklüğündeki sınır ve bir kesme geldiği an saklayıcı içeriklerinin kısa bir zaman içerisinde başka bir yere kaydedilmesi zorunluğu saklayıcı sayısının küçük tutulmasını gerektirir. Buna karşın ROM bellek

kullanımı ve sınırlı RAM, veri ve adreslerin içlerinde saklanabileceği geçici saklayıcılar gereğini doğurur. Dolayısıyla da saklayıcı sayısı ve türü konusunda bir orta yol bulmak gerekir.

Mikroişlemcilerin sözcük uzunluğunun kısa olması adresleme işlemlerini zorlaştırır. Bu zorluk aşağıda sıralanan yöntemlerle ortadan kaldırılabilir.

- Değişik saklayıcı uzunluğu. Saklayıcılardan bazılarının (program sayıcısı, bellek adres saklayıcısı, yığın göstericisi, indis saklayıcısı) uzunluğu mikroişlemcinin sözcük uzunluğundan daha büyük yapılır.
- Saklayıcı çiftlerinin beraber kullanımı. Birçok mikroişlemcide bazı saklayıcılar tek tek veya adres ve uzun veriler için bir çift olarak beraberce kullanılabilir.
- Uzun indis saklayıcıları. Bazı mikroişlemcilerde bir adres saklayıcısı gibi kullanılabilen bir indis saklayıcısı vardır. Buradan adresin tamamı saklanabilir. İndisli komutlarda bu adresten ne kadar ileri veya geri gidileceğini gösteren bir "öteleme" (offset) bilgileri bulunur.

Mikroişlemci mimarisinin bilgisayar mimarisinden ayrıldığı diğer bir nokta aritmetik-lojik birimdir. Mikroişlemcilerde ALU sadece basit aritmetik ve lojik işlemleri yapabilecek bir yapıda olup genellikle çarpma ve bölmenin bile yazılımla gerçekleştirilmesi yoluna gidilir. ALU'da genellikle tek bir taşıt vardır, operandlarından biri bir akümülatörden diğeri ise bir geçici saklayıcıdan alınır ve sonuç yine akümülatöre yerleştirilir. Buna karşın mikroişlemcilerin çoğunda bazı görevleri yerine getirebilecek özel devreler veya özel bir ROM bulunur. Ondalık (BCD) toplama ve çıkarma ROM'u bir örnek olarak verilebilir.

1.9. Mikroişlemci mimarisine örnekler:

Aşağıda iki temel mikroişlemci ailesinin (INTEL ve MOTOROLA) mimarisine birer örnek verilip açıklanmıştır. INTEL ailesine örnek olarak verilen 8080'in günümüzde artık yerini 8085'e terk etmiş olmasına rağmen tarihi bir önemi vardır. MOTOROLA ailesine örnek olarak verilen 6800 ise daha sonraları daha ayrıntılı olarak incelenecektir.

1.9.1. Intel 8080 mikroişlemci mimarisi:

Şekil 1.12'de Intel 8080'in öbek şeması verilmiştir. Sağda saklayıcı dizisi ve adres tamponu, ortada ise veri taşıtı tamponu, komut saklayıcısı ve komut çözücü devreler gösterilmiştir.

Aritmetik ve lojik işlem devreleri, bayraklar, BCD işlemler için ondalık ayarlayıcı ROM, akümülatör ve geçici saklayıcılar solda yer almaktadır.

Intel 8080'in saklayıcıları şunlardır:

- B, C, D, E, H ve L olarak adlandırılan herbiri 8 bitlik 6 genel amaçlı saklayıcı.
- 16 bitlik bir yığın göstericisi.
- 2 adet 8 bitlik geçici saklayıcı (W ve Z)

Genel amaçlı 6 saklayıcı ikişer 16 bitlik 3 saklayıcı olarak da kullanılabilir. Bu durumda B, D ve H daha önemli bayt bilgiyi taşır ve saklayıcı çiftleri bu adları taşırlar.

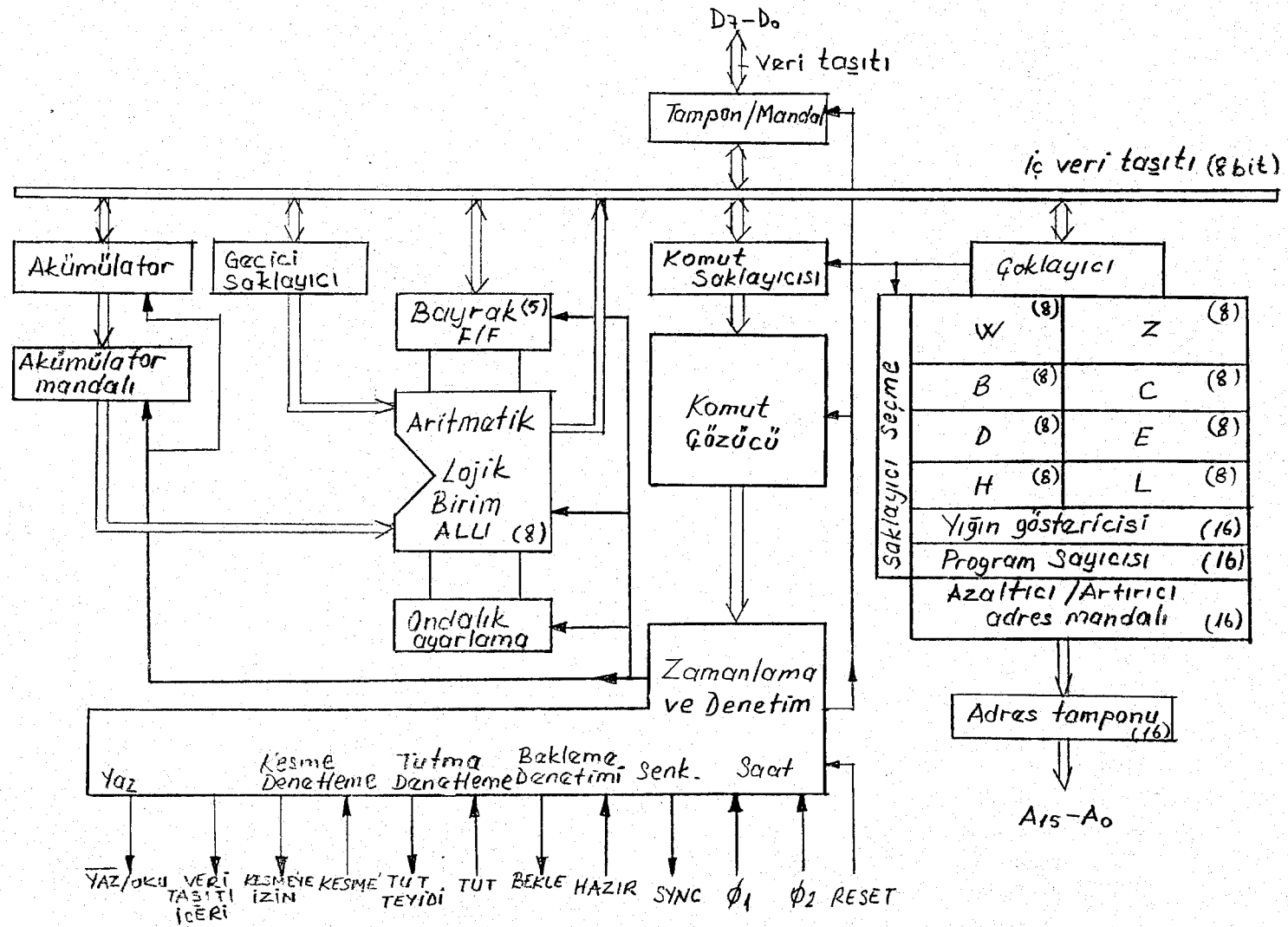
Saklayıcı çifti H (H ve L saklayıcıları) bir birincil bellek adres saklayıcısı görevini görür. Bu saklayıcı çiftinin gösterdiği bellek konumu bir genel amaçlı saklayıcı olarak kullanılabilir. Yalnız MİB'nin buraya ulaşması daha uzun bir zaman alır. Diğer saklayıcı çiftleri de birer bellek adres saklayıcısı olarak kullanılabilir. Yalnız MİB'nin buraya ulaşması daha uzun bir zaman alır. Diğer saklayıcı çiftleri de birer bellek adres saklayıcısı olarak kullanılabilirler. Fakat, bunların gösterdiği bellek konumlarındaki verinin üzerinde yapılabilecek işlemler sınırlıdır, yalnız akümülatöre ve akümülatörden yükleme yapılabilir.

Saklayıcıların kendi aritmetik kolaylıkları vardır, gerek saklayıcı çiftleri, gerekse yığın göstericisi ve program sayıcısı üzerinde W ve Z saklayıcıları aracılığı ile akümülatör ve p program sayıcısı üzerinde W ve Z saklayıcıları aracılığı ile akümülatör ve ALU kullanımı gerekmeden çeşitli işlemler yapılabilir, örneğin bir arttırılabilir veya azaltılabilirler.

16 bitlik yığın göstericisi ve program sayıcısı üzerinde W ve Z saklayıcıları aracılığı ile akümülatör ve ALU kullanımı gerekmeden çeşitli işlemler yapılabilir, örneğin bir arttırılabilir veya azaltılabilirler.

16 bitlik yığın göstericisi RAM yığınının en sön bellek konumunun adresini (en düşük adresini) içerir. Yığından bir bilgi alınmasından sonra gösterici otomatik olarak bir artar veya yığına bir bilgi itilmesin en önce azalır. Yığın büyük bir adresten küçük bir adrese doğru büyür.

8080'de komutlar 8 bitlik komut saklayıcısı ve komut çözücüsünde işlenir. Komutlar komut saklayıcısına bir tampon aracılığı ile veri taşıtına yüklenir.



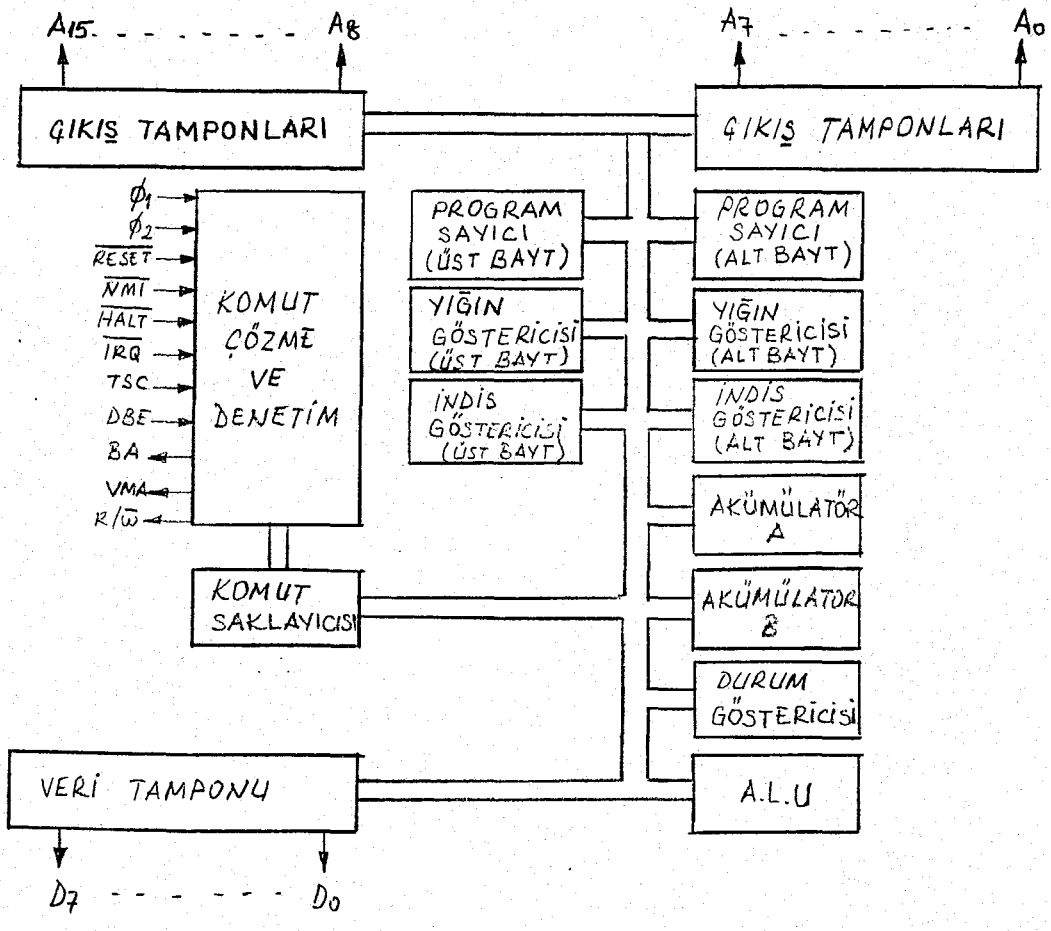
Şekil 1. 12. Intel 8080'in öbek şeması [Intel, Microprocessor data book]

Aritmetik ve lojik işlemler 8 bitlik bir ALU, bir ondalık ayarlama ROM'u, beş bayrak flip-flopu, bir akümülatör (saklayıcı A) ve bir geçici saklayıcıdan oluşan bölümde yapılır. ALU'da toplama, çıkarma ve 4 temel lojik işlemi (VE, VEYA, VAR VEYA ve DEĞİL işlemleri) ile kaydırma gerçekleştirilebilir. Bir operand daima akümülatörden, diğeri ise geçici saklayıcıdan alınır ve sonuç akümülatöre yerleştirilir. Geçici saklayıcı 6 genel amaçlı saklayıcı, 6 genel amaçlı saklayıcının herhangi birinden veya H L çiftinin gösterdiği bellekten 8 bitlik iç taşıt aracılığı ile yüklenebilir. Ondalık ayarlama ROM'u BCD toplama işlemlerinde gerekli düzeltmeleri yapmakta kullanılır.

8080'deki bayraklar ELDE, SIFIR, İŞARET, PARİTE (çift) ve YARIM ELDE (auxiliary veya half carry) bayraklarıdır. İkinin tümlevi aritmetik işlemleri için bir TAŞMA bayrağı yoktur.

1.9.2. Motorola 6800 mikroişlemci mimarisi:

Şekil 1.13'te Motorola 6800'ünü öbek şeması gösterilmiştir. Bu mikroişlemcideki saklayıcılar aşağıda sıralanmıştır.



Şekil 1.13. Motorola 6800'ün öbek şeması [Motorola, Microproccer data book]

- İki 8 bitlik akümülatör (A ve B)
- Bir 16 bitlik indis saklayıcısı
- Bir 16 bitlik program sayıcısı
- Bir 8 bitlik durum göstericisi (Condition code register)
- İki 8 bitlik geçici saklayıcı (ALU içinde yer alırlar ve öbek şemada gösterilmişlerdir).

16 bitlik indis saklayıcısı bir adres saklayıcısı gibi kullanılır. Bir baytlık bir "öteleme" bilgisi ile birlikte istenilen adrese ulaşılabilir.

8080'den farklı olarak 6800'de yığın göstericisi RAM yığında sürekli olarak boş bir konumu gösterir, yığına bir veri itildikten sonra gösterici bir azaltılır veya yığından bir veri alınmadan önce bir arttırılır. Yığın yine büyük bir adresten küçük bir adrese doğru büyür.

Komutların işlendiği bölümde bir komut saklayıcısı ve bir komut çözücüsü vardır. Komutlar, komut saklayıcısına veri taşıtından bir tampon aracılığı ile yüklenir.

Aritmetik işlemler bölümünde 8 bitlik bir ALU ve 6 bayrak yer alır. Veri ALU'ya bir geçici saklayıcıdan geçmeden doğrudan yüklenir. Akümülatör A ve B'den herhangi biri bir operand olabilir, sonuç o operandın geldiği akümülatöre yüklenir.

6800'ün 8 bitlik durum göstericisinin en önemli iki biti (bit 7 ve bit 6) sürekli 1 durumdadır, diğer 6 bit 5'ten bit 4'e doğru sıra ile YARIM ELDE, KESME, EKSİ, SIFIR, TAŞMA ve ELDE bayraklarıdır. Bu bayrakların hangi durumlarda 1, hangi durumlarda 0 olacağı daha önce açıklanmıştı.

Motorla 6800 mikroişlemci daha ileride daha ayrıntılı olarak incelenecek ve şekil 1.11'de komut çözme ve denetim öbeğine giren herbir işaret ayrı ayrı açıklanacaktır.

BÖLÜM 2
MİKROİŞLEMCİLERDE YAZILIM İLKELERİ

2.1. Giriş:

Bir mikrobilgisayar sistemi donanım ve yazılım olmak üzere iki ana parçadan oluşur. Bu bölümde sistemin yazılım parçası incelenecektir.

Mikrobilgisayar sisteminin yürüteceği program önce programı yazan kişiye kolaylık sağlayacak biçimde yazılır, sonra da mikroişlemcinin anlayabileceği bir biçime sokularak bellekte depolanır. Mikroişlemcinin program sayıcısı, program başlangıç adresi ile yüklenir ve mikroişlemci bellekteki kodları birer birer okuyarak kodun gerektirdiği işlemleri yapar. Böylece program sonuna kadar yürütülür.

2.2. Programlama dilleri:

Programların yazılımcıya doğal olan bir dille yazılmasının programlama işlemini çok kısaltacağı açıktır, fakat böyle bir dil mikroişlemci için bütünüyle anlamsızdır. Mikroişlemcinin anlayabileceği tek dil makine dilidir. Makina dilinde komutlar 01001100 gibi ikili kodlardan oluşur ve mikroişlemci komut kümesi adı verilen belirli bir kodlar grubunu tanıyabilecek şekilde tasarlanıp imal edilir.

01001100 gibi bir kodun açık bir anlamı olmadığı için makine dili insanların kullanımına uygun bir dil değildir. Bu birler sıfırlar dizisi yerine onun heksadesimal eşdeğeri olan 4C'yi kullansak bile bu da bize komutun anlamı hakkında bir fikir veremez. Atılabilecek diğer bir adım herbir komut kodu için bir sözsel simge (mnemonic) kullanmaktır. Örneğin 4C kodu 6800 mikroişlemci için A saklayıcısının bir arttırılması (increment the A register) anlamına gelir ve bu kod için INC A simgesi kullanılır. Simgeler makina kodlarına kıyasla çok daha kolayca hatırlanabilirler. Böylece programlar makina dili yerine simgesel dil (mnemonic language) kullanılarak rahatça geliştirilebilir ve sonradan makina diline çevrilebilir. Simgeler kullanarak yazılmış bir program birleştirici dil (assembly language) olarak bilinir ve bu şekilde yazılmış programlar işlenmeden önce bir çevirici (Assembler) tarafından makina diline çevrilir.

Makina dili mikroişlemcinin tasarımı sırasında belirlenir ve bir daha değiştirilmez. Birleştirici dil simgeleri ise yapımcı tarafından kullanıcıya kolaylık olmak üzere seçilmişlerdir. Örneğin INC A yerine INR A da yazılabilirdi, yeter ki çevirici

her iki simgeyi de 4C olarak makina diline çevirsin.

Birleştirici dille yazılmış programlarda makina komutlarına ek olarak çevirici program tarafından kullanılacak bazı işaretler ve komutlar da bulunur. İşaretlere örnek olarak sayıların önüne konulan $ veya % işaretlerini verebiliriz. Bunlar 6800 mikroişlemci ailesi için hazırlanmış çevirici programlarda kendilerini takip eden sayıların hiçbir işaret yoksa çevirici programlarda kendilerini takip eden sayıların heksadesimal veya ikili tabanda yazılmış olduklarını gösterir. Sayı önünde hiçbir işaret yoksa çevirici oksayıyı ondalık bir sayı olarak alır ve makina diline o şekilde çevirir. $50, $(50)_{16}$, %01001100 ise $(4A)_{16}$ anlamına gelir.

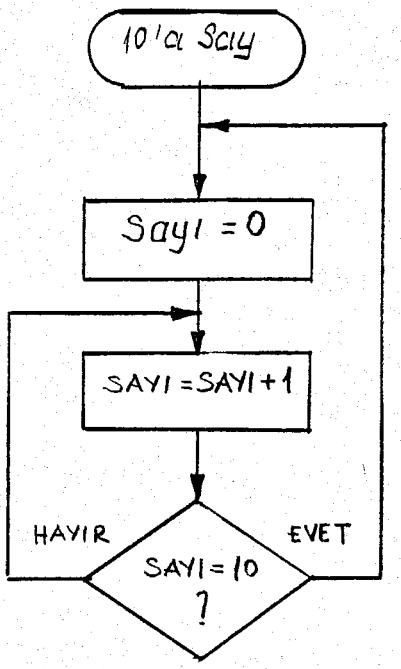
Makina diline çevrilmeyen ve yalnız çevirici program tarafından kullanılan komutlar "sözde komut" (pseudoinstructions) olarak bilinirler. Örneğin ORG A sözde komutu makina diline çevrilecek programın bellekte nereye yerleştirileceğini belirler.

Birleştirici dilde program yazmak makina dilinde programların birleştirici dilde hazırlanması yine de çok zor olur. Dolayısıyla da üst düzey diller geliştirilmiştir. Bunlar genellikle konuşulan İngilizceye benzer ve mikroişlemci türüne bağlı değildirler. Ust düzey bir dildeki tipik bir komut LET COUNT = 10 veya PRINT COUNT olabilir. Bu komutlar mikroişlemcinin yorumlayabileceğinden çok daha karmaşık olup makina diline çevrilebilmeleri için mikroişlemci belleğinde sürekli olarak saklı olan, ve derleyici (compiler) olarak bilinen uzun ve karmaşık bir programın kullanılması gerekir. Tek bir üst düzey komutu, örneğin LET COUNT = 10, makina diline çevrildiği zaman ortaya çok sayıda (bazı komutlar için düzinelerce) makina dili komutu çıkabilir.

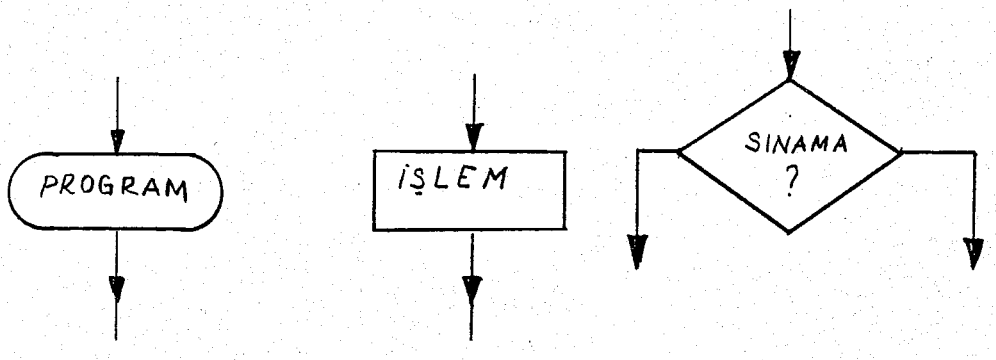
Yukarıda açıklananların daha iyi kavranabilmesi için basit bir programlama örneğini ele alalım. Akış çizeneği Şekil 2.1'de verilen program bir bellek konumunun içeriğini 0'dan 10'a kadar arttırıp aynı işlemi tekrarlamaktadır.

2.3 Akış çizenekleri:

Akış çizenekleri bir programın çalışma ilkesini grafik olarak göstermekte kullanılır. Çeşitli tür öbeklerin hatlarla birleştirilmesi ile oluşturulurlar. Akış çizeneklerinde temel olarak üç tür öbek kullanılır ve bunlar Şekil 2.2'de gösterilmiştir. Dikdörtgen öbek programın gerçekleştirildiği bir işlemi (hareketi) gösterir. Baklava dilimi öbek, bir değişkenin değerini sınama gibi işlemler için kullanılır. Gösterilen diğer öbek programın başlangıcını ve sonunu belirtir.



Şekil 2.1. Ona kadar sayma akış çizeneği.



Şekil 2.2. Akış çizeneklerinde kullanılan semboller.

Akış çizeneği program listesindeki aynı bilgileri grafik bir biçimde gösterir. Bir programın geliştirilmesindeki ilk adım genellikle, düşüncelerimizi mantıksal bir diziye oturtacak bir akış çizeneğinin çizilmesidir.

2.4. Üst-düzey dil:

Şekil 2.1'de verilen akış çizeneğinin üst düzey bir dile çevrilmesi oldukça basit bir işlemdir. BASIC üst düzey dilinde program Tablo 2.1'deki gibi olacaktır.

Satır no	Komut	Yorum
1	LET SAYI = 0	Sayıyı sıfırla
2	LET SAYI = SAYI + 1	Sayıyı 1 arttır
3	IF SAYI = 10 THEN 1	Sayı 10'sa 1'e dön
4	GO TO 2	Yoksa 2'ye dön

Tablo 2.1. Ona kadar saymak için BASIC programı.

Programın ilk iki satırı akış çizeneğinin ilk iki işlem öbeğinde belirtilenlerle aynıdır. Üçüncü ve dördüncü satırlar karar öbeğinin işlevini görmektedir (akış çizeneklerindeki EVET hattı), eşit değilse bu satırın hiçbir etkisi olmaz ve dördüncü satır, programın, sayının yine bir artırılması için ikinci dönmesini sağlar.

2.5. Birleştirici dil:

Daha önce de değinildiği gibi herbir mikroişlemcinin kendisine özgü bir makina dili, dolayısıyla da bir birleştirici dili vardır. Yukarıda açıklanan örnek program 6800 mikroişlemcisinin birleştirici dilinde Tablo 2.2'de gösterilen biçimde olacaktır. Bu program BASIC programına kıyasla çok daha şifreli bir görünüme sahip olmasına rağmen sonuçta aynı işlemi görmektedir. Birleştirici dilin özellikleri ile mikroişlemcinin yapısal özellikleri arasında doğrudan bir bağlantı olduğundan bu şekilde yazılmış bir programın İngilizce'ye yakınlığı daha az olmamak zorundadır.

	Komut		
Etiket	İşlem kodu	Operand	Yorum
BAŞLA	CLR A		Akümülatör A'yı sıfırla
DÖNGÜ	INC A		A'yı bir artır
	CMP A	# 10	A ile 10'u karşılaştır
	BEQ	BAŞLA	A = 0 ise başa dön
	JMP	DÖNGÜ	Tekrarla

Tablo 2.2. Birleştirici dilde program.

Tablo 2.2'de etiket, komut ve yorum olmak üzere üç kolon gösterilmiştir. Etiket BASIC dilindeki satır sayısına benzer. Her satırı numaralandırmak yerine, program içerisinde işaret edilmesi gereken satırlara birer etiket konulur. Komut kolonu, işlem kodu (op-code) ve operand olmak üzere iki ayrı alandan oluşur. Operand alanında işlem kodunun üzerinde gerçekleştirileceği operand yer alır. Bazı işlem kodları (CLR A, INC A gibi) operand gerektirmezler. Buna karşı A akümülatörünü belirli bir sayı ile karşılaştırmaya yarayan CMP A işlem kodu için A'nın ne ile karşılaştırılacağını operand belirtmek gerekir. 10 sayısının önündeki işaret daha sonra da açıklanacağı gibi A'nın hemen 10 sayısı ile (Örneğin 10 numaralı bellek konumunun içeriği ile değil) karşılaştırılacağını gösterir (hazır adresleme-immediate addressing).

Yorum kolonunda ise program açıklamaları bulunur. Bunlar gerek programın kullanıcısına gerekse, eğer programı hazırlayan kişinin bir zaman sonra aynı program üzerinde tekrar çalışması gerekirse o kişiye, program hakkında açıklayıcı bilgiler verirler. BASIC dilinde dilin kendisi açıklayıcı olduğundan fazla bir yorum gerekmeyebilir, fakat birleştirici dilde yorumlar vazgeçilmez birer unsurdur.

Programı açıklayacak olursak ilk komut CLR A (Clear A) A akümülatörünü sıfırlar. Bu komut BASIC'deki LET SAYI = 0 komutuna eşdeğerdir, tek fark değişken için bir ad (SAYI) kullanılacağına birleştirici dilde A akümülatörü değişken olarak tanımlanmıştır.

INC A (Increment A) akümülatör A'daki sayıyı bir artırır anlamına gelir. Dolayısıyla LET SAYI = SAYI + 1 BASIC komutuna eşdeğerdir. INC A komutunun işlevi kısaca A ← (A) + 1 şeklinde gösterilir. Burada parantez içerik amlamını verir, ok ise A'nın içeriğine 1 eklenmesinde elde edilen sonucunun tekrar A'ya yerleştirildiğini belirtmektedir. Bu tür gösterimleri ileride sık sık kullanacağız.

Daha sonra gelen üç komut karar işlevini yerine getirir. CMP A #10 (compare A with 10) komutunun yaptığı tek işlem A'nın içeriğinden A'yı değiştirmeden 10 çıkarmaktır ve sonuç ne olursa olsun tek başına bir atlamaya neden olmaz. Fakat sonuç sıfır ise SIFIR bayrağı bir durumuna geçer. BEQ (branch if equal) komutu bu bayrağa bakar ve eğer bayrak 1 durumunda ise (karşılaştırma işleminin sonucu sıfır çıkmışsa) BAŞLA etiketini taşıyan satıra atlamaya neden olur. Aksi taktirde bir sonraki komut olan JMP (jump) komutuna geçirilir. O halde CMP A #10 ve

BEQ DÖNGÜ komutları BASIC'teki IF SAYI = 10 THEN 1 komutuna eşdeğerdir. Son komut JMP DÖNGÜ komutu ise BASIC'deki GO TO 2 komutu gibi koşulsuz bir atlama gerçekleştirir.

2.6 Makina dili:

Tablo 2.2'deki birleştirici dil programının makina diline çevrilmiş biçimi programın başlangıç adresinin $(07A0)_H$ olduğu (CLR A komutunun üstünde ORG $07A0 komutun olduğu) varsayılarak Tablo 2.3'de verilmiştir. Fakat bu iki tablodaki programların aynı olduğunu ilk bakışta anlamak bir uzman için bile zordur. Tablo 2.3'de verilen programın işlevi birler sıfırlar dizilerinin ardında tümüyle kaybolmuştur. Bu da bize makina dilinin zorluklarını apaçık göstermektedir. Kullanımının tek nedeni bu dilin mikroişlemcinin anlayabileceği tek dil olmasıdır.

Bellek adresi (Heks)	Bellek içeriği (Heks)	(İkili)
07A0	4F	01001111
07A1	4C	01001100
07A2	81	10000001
07A3	0A	00001010
07A4	27	00100111
07A5	FA	11111010
07A6	7E	01111110
07A7	07	00000111
07A8	A1	10100001

Tablo 2.3. Ona kadar sayma programının makina dilindeki biçimi

Makina dilinde yazılmış programı açıklayabilmek için programın üç ayrı dilde yazılmış şeklini bir arada gösterip karşılaştıran Tablo 2.4'e bakalım. Her bellek konumunda sekiz bitlik bilgi saklanabilmektedir. Herbir komut yapılacak işlemi gösteren bir işlem kodu ile başlamaktadır. 6800'de işlem kodları birer baytlık olup işlemin cinsine göre arkalarından sıfır, bir, veya iki baytlık bilgi gelmesi gerekir.

07A0 sayılı bellek konumunun içeriği programın ilk baytı 4F olup bu CLR komutunun işlem kodudur. Yapılacak işlem başka

bir bilgi gelmesi gerekmez. Dolayısıyla bir sonraki bellek konumundaki bilgi, yani 07A1'deki 4C yine bir işlem kodu olup akümülatör A'nın bir artması sağlar (INC A). Bu konut da başka bir bilgi gerektirmeyeceğine göre o da bellekte tek baytlık yer kaplayan bir komut türüdür.

07A2'deki 81 işlem kodu akümülatör A'nın bir sayıyla karşılaştırılacağı anlamını verir. Dolayısıyla da karşılaştırılacak sayının bir sonraki bellek gözünde yer alması gerekir. Yani CMP A #10 bellekte iki baytlık bir yer alır.

Daha önce de değinildiği gibi CMP komutu bayrak etkileyen bir komuttur, eğer A = 10 ise ZERO bayrağı kalkar. 07A4'deki 27 işlem kodunun gösterdiği BEQ komutu bu bayrağı sınar ve eğer bayrak birse bir dallanmaya neden olur (BRANCH IF EQUAL-EŞİTSE DALLAN). Dallanılacak yer ise bir sonraki bellek gözündeki "öteleme" bilgisinin program sayıcısına eklenmesi ile bulur. Öteleme (offset) bilgisi işaretli bir sayı olup +127 ile -128 ($7F ile $80) arasında herhangi bir sayı olabilir. 07A5 sayılı bellek gözündeki FA verisinin mikroişlemci tarafından veri taşıtında veri taşıtından içeriye alınması ile program sayıcının otomatik olarak bir arttığına daha önce değinilmişti. Dolayısıyla da dallanılacak adres FA'nın 07A6'ya eklenmesi ile 07A0 olarak bulunur. Eğer ZERO bayrağı 0 ise hiçbir işlem yapılmaz ve program sayıcısı 07A6 olarak kalır.

Makina diline çevrilecek en son komut bir ATLA (JMP) komutudur. Bu komutun işlem kodu 7E'dir ve arkasından hemen atlanacak adresin alt baytının bellekte daha önce, üst baytının ise daha sonra yer aldığını belirtmek yerinde olacaktır. Örneğin 8080'in (veya 8085'in) birleştirici dilinde son komut JZ DÖNGÜ olarak yazılacak ve makina dilinde ise bu C3 A1 07 olarak kodlanacaktı.

Görülüyor ki makina dilinde yazılmış bir program bir baytlar dizisinden oluşmaktadır. Herbir baytın üç ayrı anlamı olabilir. Bazıları işlem kodudur, bazıları üzerinde işlem yapılacak veridir (CMP A #10:81 0A daki 0A gibi), bazıları ise bir adres parçasıdır (7E 07 Altbaytlar dizisindeki 07 veya A1 gibi). Mikroişlemci içerisindeki devreler aracılığı ile bir işlem kodu çözüldüğü zaman arkasından kaç baytlık ve ne tür bir bilgi geleceği anlaşılır ve böylece program istenilen şekilde işlenir.

Tablo 2.4. Ona kadar sayma programının üç dilde yazılış şekli ve karşılaştırması.

BASIC DİLİ		6800 BİRLEŞTİRİCİ DİLİ			6800 MAKİNA DİLİ	
Satır	Komut	Etiket	Komut	Adres	İçerik	Açıklama
1	LET SAYI=0	BAŞLA	CLR A	07A0	4F	İşlem kodu
2	LET SAYI=SAYI+1	DÖNGÜ	INC A	07A1	4C	İşlem kodu
3	IF SAYI=10 THEN1		CMP A #10	07A2	81	İşlem kodu
				07A3	0A	Veri
			BEQ BAŞLA	07A4	27	İşlem kodu
				07A5	FA	Adres için Veri
4	GO TO 2		JMP DÖNGÜ	07A6	7E	İşlem Kodu
				07A7	07	Adres
				07A8	A1	

2.7. Sonuç:

Bu bölümde açıklananları özetleyecek olursak üst düzey diller programcıya birçok kolaylıklar sağlar ve her tür mikrobilgisayarda kullanılabilir. Fakat mikrobilgisayarın belleğinde, programları makina koduna çevirecek uzun bir derleyicinin yer alması gerekir. Ayrıca yüksek düzey diller çalışma hızları ve bellek gereksinimleri açısından oldukça verimsizdirler. Birleştirici dilde yazılmış bir program üst düzey bir dilde yazılarak bir derleyici ile makina diline çevrilmiş eşdeğerine kıyasla bellekte çok daha az yer kaplar ve daha hızlı çalışır.

Birleştirici dil, programcı açısından üst-düzey dillere kıyasla çok daha zor olmasına rağmen, mikroişlemcilerin programlanmasında, özellikle hızlı çalışma ve az bellek gerektiren uygulamalarda, yaygın olarak kullanılır.

Makina dili, mikroişlemcinin doğrudan anlayabildiği tek dil olmasına karşın, insanların kullanımına uygun değildir. Programlar genellikle birleştirici dilde yazılır ve bir çevirici (assembler) kullanılarak makina diline dönüştürülür.

BÖLÜM 3
6800 AİLESİ MİKROİŞLEMCİLERİ KOMUT KÜMESİ

3.1. Giriş:

6800 mikroişlemci komut kümesi 1, 2 veya 3 baytlık 72 komut içerir. Bir komutun uzunluğu komuta ve aşağıda açıklanacak adresleme çeşidine bağlı olup ilk(veya tek) baytı komutu ve kullanılan adresleme çeşidini belirlemeye yeterlidir. 72 komutun bütün geçerli adresleme çeşitleri için heksa değerleri Tablo 3.1'-de verilmiştir. Kullanılabilecek 256 heksadesimal sayıdan 197'sinin geçerli birer makina kodu (machine code) olduğu, 56'sının ise kullanılmadığı bu tablodan görülebilir.

Eğer bir komut iki veya üç baytdan oluşursa, ikinci veya ikinci ve üçüncü baytlar bir operand, bir adres veya adres elde etmekte kullanılacak bilgiyi içerir.

6800 mikroişlemcinin komut kümesi

1. Akümülatör ve bellek operasyonları
2. Program denetleme operasyonları
3. Durum Saklayıcısı operasyonları

olmak üzere üç genel gruba ayrılabilir. 6800'de giriş/çıkış operasyonlarının bellek operasyonlarından farksız olduğuna daha önce değinilmişti. Dolayısıyla da giriş/çıkış için özel komutlar yoktur.

Aşağıda önce adresleme çeşitleri incelenmiş daha somra da bu üç grup operasyon için kullanılan komutlar açıklanmıştır.

3.2 Adresleme çeşitleri:

Bölüm 4.1.'de 6802 tanıtılırken bu mikroişlemcinin hazır (immediate), doğrudan (direct), genişletilmiş (extended), içerilmiş (inherent, implied), indisli (indexed), bağıl (relative) ve akümülatör adresleme olmak üzere 7 adresleme çeşidi olduğuna değinilecektir. Bunlardan içerilmiş ve akümülatör adresleme benzer bir adresleme çeşidi olduğundan aşağıda aynı başlık altında açıklanmıştır.

3.2.1. Hazır (hemen) adresleme (immediate addressing):

Bu tür adresleme kullanan komutların çoğunluğu iki bayttan oluşur. Birinci bayt işlem kodudur, ikinci bayt ise operand veya kullanılacak bilgidir. Örneğin A akümülatöründe $2C olduğu varsayarak aşağıda belirtilen komut $23'ün akümülatöre eklenmesine neden olur.

Tablo 3.1. Makina kodlarının hexadesimal eşdeğerleri
[Motorola, Microprocessor data book]

00	•		40	NEG	A		80	SUB	A	IMM	C0	SUB	B	IMM
01	NOP		41	•			81	CMP	A	IMM	C1	CMP	B	IMM
02	•		42	•			82	SBC	A	IMM	C2	SBC	B	IMM
03	•		43	COM	A		83	•			C3	•		
04	•		44	LSR	A		84	AND	A	IMM	C4	AND	B	IMM
05	•		45	•			85	BIT	A	IMM	C5	BIT	B	IMM
06	TAP		46	ROR	A		86	LDA	A	IMM	C6	LDA	B	IMM
07	TPA		47	ASR	A		87	•			C7	•		
08	INX		48	ASL	A		88	EOR	A	IMM	C8	EOR	B	IMM
09	DEX		49	ROL	A		89	ADC	A	IMM	C9	ADC	B	IMM
0A	CLV		4A	DEC	A		8A	ORA	A	IMM	CA	ORA	B	IMM
0B	SEV		4B	•			8B	ADD	A	IMM	CB	ADD	B	IMM
0C	CLC		4C	INC	A		8C	CPX	A	IMM	CC	•		
0D	SEC		4D	TST	A		8D	BSR		REL	CD	•		
0E	CLI		4E	•			8E	LDS		IMM	CE	LDX		IMM
0F	SEI		4F	CLR	A		8F	•			CF	•		
10	SBA		50	NEG	B		90	SUB	A	DIR	D0	SUB	B	DIR
11	CBA		51	•			91	CMP	A	DIR	D1	CMP	B	DIR
12	•		52	•			92	SBC	A	DIR	D2	SBC	B	DIR
13	•		53	COM	B		93	•			D3	•		
14	•		54	LSR	B		94	AND	A	DIR	D4	AND	B	DIR
15	•		55	•			95	BIT	A	DIR	D5	BIT	B	DIR
16	TAB		56	ROR	B		96	LDA	A	DIR	D6	LDA	B	DIR
17	TBA		57	ASR	B		97	STA	A	DIR	D7	STA	B	DIR
18	•		58	ASL	B		98	EOR	A	DIR	D8	EOR	B	DIR
19	DAA		59	ROL	B		99	ADC	A	DIR	D9	ADC	B	DIR
1A	•		5A	DEC	B		9A	ORA	A	DIR	DA	ORA	B	DIR
1B	ABA		5B	•			9B	ADD	A	DIR	DB	ADD	B	DIR
1C	•		5C	INC	B		9C	CPX		DIR	DC	•		
1D	•		5D	TST	B		9D	•			DD	•		
1E	•		5E	•			9E	LDS		DIR	DE	LDX		DIR
1F	•		5F	CLR	B		9F	STS		DIR	DF	STX		DIR
20	BRA	REL	60	NEG		IND	A0	SUB	A	IND	E0	SUB	B	IND
21	•		61	•			A1	CMP	A	IND	E1	CMP	B	IND
22	BHI	REL	62	•			A2	SBC	A	IND	E2	SBC	B	IND
23	BLS	REL	63	COM		IND	A3	•			E3	•		
24	BCC	REL	64	LSR		IND	A4	AND	A	IND	E4	AND	B	IND
25	BCS	REL	65	•			A5	BIT	A	IND	E5	BIT	B	IND
26	BNE	REL	66	ROR		IND	A6	LDA	A	IND	E6	LDA	B	IND
27	BEQ	REL	67	ASR		IND	A7	STA	A	IND	E7	STA	B	IND
28	BVC	REL	68	ASL		IND	A8	EOR	A	IND	E8	EOR	B	IND
29	BVS	REL	69	ROL		IND	A9	ADC	A	IND	E9	ADC	B	IND
2A	BPL	REL	6A	DEC		IND	AA	ORA	A	IND	EA	ORA	B	IND
2B	BMI	REL	6B	•			AB	ADD	A	IND	EB	ADD	B	IND
2C	BGE	REL	6C	INC		IND	AC	CPX		IND	EC	•		
2D	BLT	REL	6D	TST		IND	AD	JSR		IND	ED	•		
2E	BGT	REL	6E	JMP		IND	AE	LDS		IND	EE	LDX		IND
2F	BLE	REL	6F	CLR		IND	AF	STS		IND	EF	STX		IND
30	TSX		70	NEG		EXT	B0	SUB	A	EXT	F0	SUB	B	EXT
31	INS		71	•			B1	CMP	A	EXT	F1	CMP	B	EXT
32	PUL	A	72	•			B2	SBC	A	EXT	F2	SBC	B	EXT
33	PUL	B	73	COM		EXT	B3	•			F3	•		
34	DES		74	LSR		EXT	B4	AND	A	EXT	F4	AND	B	EXT
35	TXS		75	•			B5	BIT	A	EXT	F5	BIT	B	EXT
36	PSH	A	76	ROR		EXT	B6	LDA	A	EXT	F6	LDA	B	EXT
37	PSH	B	77	ASR		EXT	B7	STA	A	EXT	F7	STA	B	EXT
38	•		78	ASL		EXT	B8	EOR	A	EXT	F8	ADC	B	EXT
39	RTS		79	ROL		EXT	B9	ADC	A	EXT	F9	ADC	B	EXT
3A	•		7A	DEC		EXT	BA	ORA	A	EXT	FA	ORA	B	EXT
3B	RTI		7B	•			BB	ADD	A	EXT	FB	ADD	B	EXT
3C	•		7C	INC		EXT	BC	CPX		EXT	FC	•		
3D	•		7D	TST		EXT	BD	JSR		EXT	FD	•		
3E	WAI		7E	JMP		EXT	BE	LDS		EXT	FE	LDX		EXT
3F	SWI		7F	CLR		EXT	BF	STS		EXT	FF	STX		EXT

Notlar: 1. Adresleme: A-Akümülatör A, B- Akümülatör B, IMM- Hemen, DIR- Doğrudan, REL- Bağıl, IND- İndisli.

2. Tanımlanmış kod " • " ile gösterilmiştir.

Sembolik Dilde	Makina Dili	Anlamı
ADDA # Ø 23	8B23	A ← (A) + Ø23

Bu komuttan sonra akümülatörün içeriği Ø4F olacaktır. İndis saklayıcı veya Yığın Göstericisi ile ilgili komutlarda (örneğin CPX: Compare Index Register Saklayıcısını karşılaştır, LDX: Load Index Register-Indis Saklayıcısını yükle veya LDS: Load Stack Pointer-Yığın Göstericisi yükle) hazır adresleme kullanılırsa bu saklayıcılar 16 bitlik olduğundan karşılaştırılacak veya yüklenecek sayının da iki baytlık olması gerekir. Dolayısıyla da bu tür komutlar üç bayttan oluşur.

Simgesel dilde yazılmış bir program bir çevirici ile makina diline çevrilirken, çeviriciye hazır adresleme kullandığı işareti belirtir.

3.2.2. Doğrudan adresleme (direct addressing):

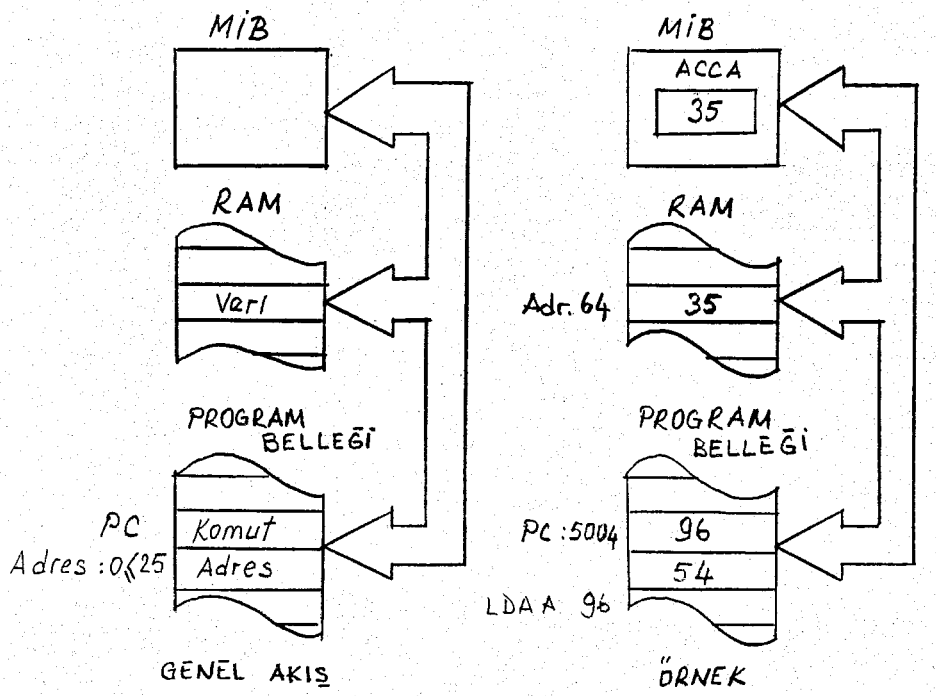
Bu tür bir adresleme kullanan komutların tümü iki bayttan oluşur. İkinci bayt üzerinde işlem yapılacak verinin adresidir. Adres sadece 8 bitten olacağına göre MİB'min adres taşıtına çıkardığı verinin üst baytı efektif olarak "0" olur. Dolayısıyla da bü tür bir adreslemede ulaşılabilecek bellek konumları Ø0000 ile Ø00FF arasında sınırlıdır. Örneğin Ø64 adresini taşıyan bellek konumunun içeriği Ø35 ise aşağıda belirtilen komut A akümülatörüne Ø35'in yüklenilmesine neden olur.

Simgesel dil	Makina dili	İşlevi
LDAA Ø64	96 64	A ← (Ø64)

Bu adresleme türünde veri akışı Şekil 3.1'de gösterilmiştir.

Burada 16 bitlik saklayıcılar için bu saklayıcılara yüklenecek verinin ikinci baytının hangi adresten elde edileceği sorusu akla gelebilir. Böyle bir durumda 16 bitlik saklayıcının üst baytına komutla gösterilen adresin içeriği, alt baytına ise bir sonraki adresin içeriği yüklenir. Aşağıda bir örnek verilmiştir.

Simgesel dil	Makina dili	İşlevi
LDS Ø64	9 E 64	SPH←(Ø64) ve SPL←(Ø65)



Şekil 3.1. Doğrudan adreslemede veri akışı.

3.2.3. Genişletilmiş adresleme (extended addressing):

Bu tür adresleme doğrudan adreslemenin genişletilmiş bir şekli olup $0000-$FFFF arasındaki bütün konumlara erişilebilmesini sağlar. Adres olarak iki bayt kullanılması gerekeceğine göre komut da üç bayttan oluşur. Örneğin $703F adresinde $46 olduğunu varsayarak aşağıdaki komut B akümülatörüne $46'nın yüklenmesine neden olur.

Simgesel dil	Makina dili	İşlevi
LDAB $703F	F6 703F	B ← ($703F)

Genişletilmiş adres kullanan komutlar, doğrudan adres kullananlara göre bellekte bir bayt daha fazla yer kapladıkları gibi yürütülmeleri için de birden fazla makina çevirimi gerektirirler. Onun için bir programda genişletilmiş adresleme yerine mümkün olduğu kadar doğrudan adresleme kullanılmalıdır. Bunu sağlayabilmek için de yaz/boz bellek, (geçici sonuçların saklanacağı bellek) Bölüm 4.5'de de değinildiği gibi, $0000-$00FF konumları arasına yerleştirilmelidir.

Genişletilmiş adresleme veri akışı Şekil 3.2'de gösterilmiştir. MİB $5006 no'lu bellek konumunda LDAB'nin işlem kodu olan F6'yı gördüğü zaman takip eden iki bellek gözündeki 703F'yi okur ve adres taşıtına bu sayıyı çıkarır ve böylece veri taşıtı aracılığı ile $46 bilgisinin B akümülatörüne alır.

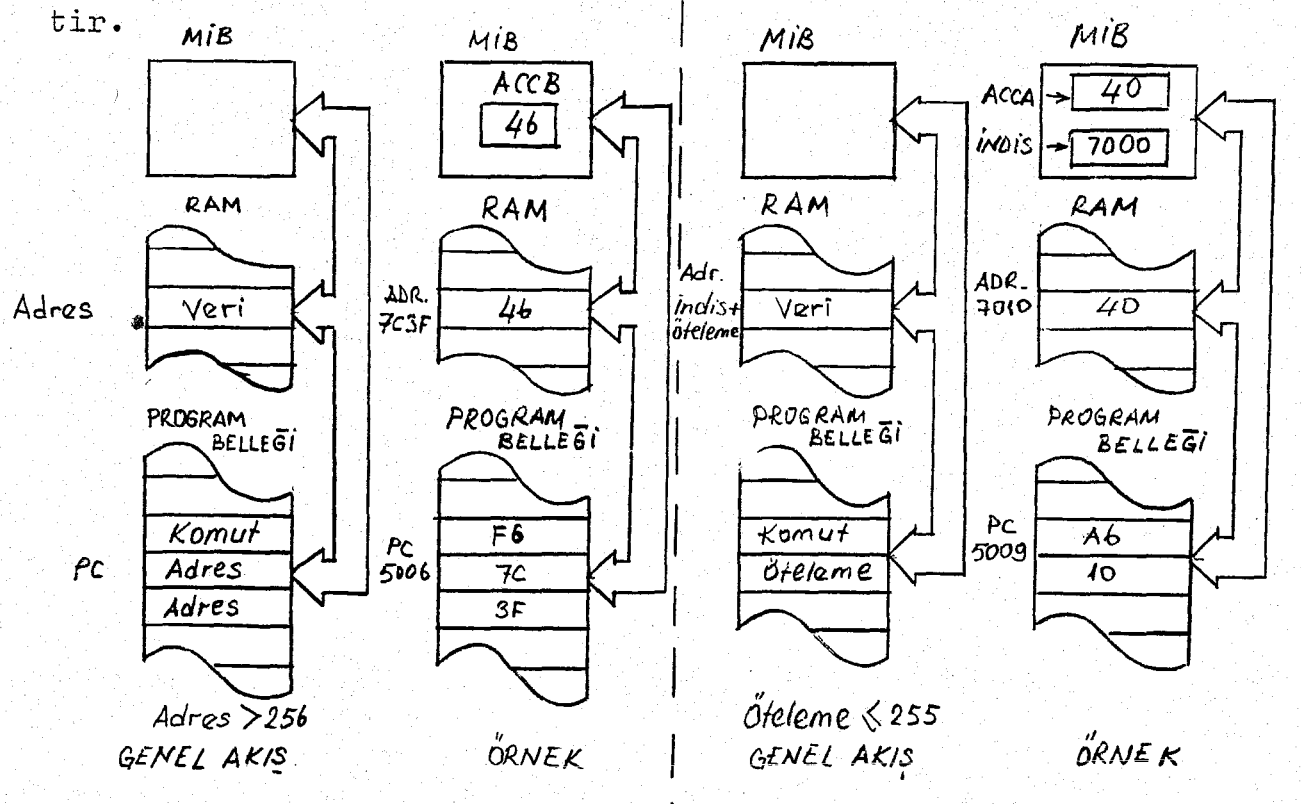
3.2.4. İndisli adresleme (Indexed addressing):

İndisli adresleme ulaşılmak istenen adres değişken olup İndis Saklayıcısının içeriğine bağlıdır. Gerekli adres programı yürütülmesi sırasında İndis Saklayıcısının yardımıyla elde edilir. (Diğer adresleme çeşitlerinde program yazılıp makina diline çevrildiği an artık adres belirlenmiştir) Komutun operand alanında Indis saklayıcısının içeriğine eklenecek öteleme sayısı (Offset) olarak bilinen bir baytlık (0-255 arası) bir sayı da olmalıdır. Böylece İndis Saklayıcısının gösterdiği adresten 255 (ØFF) kadar daha ileriye ulaşılabilir.

Bu tür adresleme kullanan komutlar iki bayttan oluşur. Birinci bayt işlem kodu ikinci bayt ise öteleme baytıdır. Eğer doğrudan doğruya indis Saklayıcısının gösterdiği adrese ulaşılacaksa ikinci bayt sıfır olur. Örneğin İndis Saklayıcısının Ø7000 olduğunu ve Ø7010 no'lu bellekte Ø40 olduğunu varsayarak aşağıdaki komut A akümülatörüne Ø40 yüklenmesine neden olacaktır.

Simgesel dil	Makina dili	İşlevi
LDAA Ø 10,X	A6 10	A ← (X + Ø10)

Yukarıdaki örnek için veri akışı Şekil 3.3'de gösterilmiştir.



ŞEKİL 3.2. Genişletilmiş adreslemede veri akışı

ŞEKİL 3.3. İndisli adreslemede veri akışı

3.2.5. Bağıl adresleme (relative addressing)

Sadece dallanma komutlarında kullanılan bu adresleme türünde ulaşılması gereken adres program sayıcısının o andaki içeriğine bağlı olarak bulunur. Dallanma komutları iki bayttan oluşur. Birinci bayt işlem baytıdır, ikinci bayt ise bir öteleme baytı olup program sayıcısına eklenir. Bir dallanma sırasında ileriye gidilmesi gerekeceği gibi geriye de gidilmesi gerekebileceğinden öteleme sayısı işaretli bir sayıdır. Böylece program sayıcısının gösterdiği yerden +127 ileriye veya -128 geriye gidilebilir. Fakat öteleme sayısı okunduğu zaman program sayıcının dallanma komutunun olduğu yerden iki ilerisini göstereceği düşünülecek olursa dallanabilecek alanın dallanma komutunun olduğu yerden +129 ileri veya -126 geri olacağı ortaya çıkar. Örnek olarak belirli bir komutun yürütülmesi sonucunda Durum Saklıcısının Z biti (sıfır bayrağı) "1" durumuna geçmişse, bulunduğumuz yerden 10 adım ileriye dallanmamız aksi taktirde programa devam etmemiz gerekliliğini varsayalım. Bu durumda kullanmamız gereken öteleme sayısı 8 olacaktır.

Simgesel dil	Makina dili Adres	İçerik	İşlevi
REF BEQ 8	REF	27	Z =1: (PC) + 8 → PC
	REF +1	08	Z =0: PC'de değişiklik yok
	REF +2	xx	

Yukarıdaki örnek programda BEQ (Branch if Equal to Zero-Sıfırsa dallan) komutu Z bayrağını sınayan bir komuttur. MİB bu komutun işlem kodu 27'yi okuduğu zaman Durum Saklayıcısının Z bitine bakar, bu bit 0'sa, öteleme sayısını da okur, fakat PC'yi değiştirmeden gösterdiği yerdeki (ki bu anda REF +2'yi göstermektedir) komutu içeriye alarak programa devam eder. Böylece PC, REF +10 konumunu gösterir duruma geçeceğinden içeriye alınacak yeni komut buradaki komut olacaktır.

Eğer 10 adım ileri yerine 20 adım geri gitmemiz gerekiyorsa, yani Z = 1 ise REF-20 konumdaki komuta dönülmek isteniyorsa öteleme sayısının -22 olması gerekir. O halde program

Simgesel dil	Makina dili Adres	İçerik	İşlevi
REF BEQ -22	REF	27	Z =1 PC←(PC) -22
	REF +1	EA	Z =0 PC'de değişiklik yok
	REF +2	xx	

şeklinde olmalıdır.

3.2.6. İçerilmiş ve akümülatör adresleme (inherent and accumulator addressing):

Bu tür adreslemede komutunun gerektirdiği bütün bilgiler MİB içerisinde var olup bellekten veya programdan herhangi bir operandın alınması gerekmez. Örneğin ABA (Add accumulator B to accumulator A - A akümülatörünü B ile topla), CLRA (Clear A - A akümülatörünü sıfırla) veya CLC (Clear Carry - Eldeyi sıfırla) komutlarında bütün gerekli bilgiler var olup ilk ikisi akümülatör adreslemeye, üçüncüsü ise içerilmiş adreslemeye birer örnektirler. Bu tür konutlar makina diline tek bir baytla çevrilirler.

3.3. Akümülatör ve bellek işlemleri:

Akümülatör ve bellek işlemleri için kullanılabilecek komutlar, simgesel gösterimleri, adresleme türleri, makina kodları, gerektirdikleri bayt ve makina çevrim sayıları, işlemleri durum göstericisi üzerindeki etkileri ile birlikte toplu halde Tablo 3.2'de gösterilmiştir. Bu tabloda (ve daha sonra verilecek tablolarda) kullanılan kısaltma ve işaretler aşağıda açıklanmıştır.

IMMED: Hazır adresleme
DIRECT: Doğrudan adresleme
INDEX: İndisli adresleme
OP: İşlem kodu (heksadesimal)
~ : MİB çevrim sayısı
# : Program bayt sayısı
+ : Aritmetik artı
− : Aretmetik eksi
· : Lojik VE
+ : Lojik VEYA
⊕ : Lojik dışlayan VEYA
• : Etkilenmez
→ : Aktarma

EXTND: Genişletilmiş adresleme
INHER: İçerilmiş adresleme
RELATIVE: Bağıl adresleme
LS: En önemsiz
MS: En önemli
CCR: Durum göstericisi
0: Bit=Sıfır
00: Bayt=Sıfır
R: Daima sıfırlanır (Reset)
S: Daima bir yapılır (set)
↕: Sınanır, doğru ise "1", aksi takdirde 0 yapılır
M_{SP}: Yığın göstericisinin işaret ettiği bellek konumu

H, I, N, Z, V ve C : Durum saklayıcısının sıra ile yarım elde, kesme maskesi, eksi, sıfır, taşma ve elde bitleri.

Tablo 3.2'de verilen işlemler kendi içlerinde (1) Aritmetik İşlemler, (2) Lojik İşlemleri, (3) Veri Sınama ve (4) Veri Üzerinde İşlemler olmak üzere 4 gruba ayrılabilirler.

3.3.1. Aritmetik İşlemler:

Tablo 3.3.'de özetlenen bu işlemler,

Tablo 3.2. Akümülatör ve bellek işlemleri

[Motorola, Microprocessor data book]

OPERATIONS	MNEMONIC	IMMED OP	IMMED ~	IMMED =	DIRECT OP	DIRECT ~	DIRECT =	INDEX OP	INDEX ~	INDEX =	EXTND OP	EXTND ~	EXTND =	IMPLIED OP	IMPLIED ~	IMPLIED =	BOOLEAN/ARITHMETIC OPERATION (All register labels refer to contents)	5 H	4 I	3 N	2 Z	1 V	0 C
Add	ADDA	8B	2	2	9B	3	2	AB	5	2	BB	4	3				A + M → A	↕	•	↕	↕	↕	↕
	ADDB	CB	2	2	DB	3	2	EB	5	2	FB	4	3				B + M → B	↕	•	↕	↕	↕	↕
Add Accmtrs	ABA													1B	2	1	A + B → A	↕	•	↕	↕	↕	↕
Add with Carry	ADCA	89	2	2	99	3	2	A9	5	2	B9	4	3				A + M + C → A	↕	•	↕	↕	↕	↕
	ADCB	C9	2	2	D9	3	2	E9	5	2	F9	4	3				B + M + C → B	↕	•	↕	↕	↕	↕
And	ANDA	84	2	2	94	3	2	A4	5	2	B4	4	3				A · M → A	•	•	↕	↕	R	•
	ANDB	C4	2	2	D4	3	2	E4	5	2	F4	4	3				B · M → B	•	•	↕	↕	R	•
Bit Test	BITA	85	2	2	95	3	2	A5	5	2	B5	4	3				A · M	•	•	↕	↕	R	•
	BITB	C5	2	2	D5	3	2	E5	5	2	F5	4	3				B · M	•	•	↕	↕	R	•
Clear	CLR							6F	7	2	7F	6	3				00 → M	•	•	R	S	R	R
	CLRA													4F	2	1	00 → A	•	•	R	S	R	R
	CLRB													5F	2	1	00 → B	•	•	R	S	R	R
Compare	CMPA	81	2	2	91	3	2	A1	5	2	B1	4	3				A − M	•	•	↕	↕	↕	↕
	CMPB	C1	2	2	D1	3	2	E1	5	2	F1	4	3				B − M	•	•	↕	↕	↕	↕
Compare Accmtrs	CBA													11	2	1	A − B	•	•	↕	↕	↕	↕
Complement, 1's	COM							63	7	2	73	6	3				M̄ → M	•	•	↕	↕	R	S
	COMA													43	2	1	Ā → A	•	•	↕	↕	R	S
	COMB													53	2	1	B̄ → B	•	•	↕	↕	R	S
Complement, 2's (Negate)	NEG							60	7	2	70	6	3				00 − M → M	•	•	↕	↕	①	②
	NEGA													40	2	1	00 − A → A	•	•	↕	↕	①	②
	NEGB													50	2	1	00 − B → B	•	•	↕	↕	①	②
Decimal Adjust, A	DAA													19	2	1	Converts Binary Add. of BCD Characters into BCD Format	•	•	↕	↕	↕	③
Decrement	DEC							6A	7	2	7A	6	3				M − 1 → M	•	•	↕	↕	④	•
	DECA													4A	2	1	A − 1 → A	•	•	↕	↕	④	•
	DECB													5A	2	1	B − 1 → B	•	•	↕	↕	④	•
Exclusive OR	EORA	88	2	2	98	3	2	A8	5	2	B8	4	3				A⊕M → A	•	•	↕	↕	R	•
	EORB	C8	2	2	D8	3	2	E8	5	2	F8	4	3				B⊕M → B	•	•	↕	↕	R	•
Increment	INC							6C	7	2	7C	6	3				M + 1 → M	•	•	↕	↕	⑤	•
	INCA													4C	2	1	A + 1 → A	•	•	↕	↕	⑤	•
	INCB													5C	2	1	B + 1 → B	•	•	↕	↕	⑤	•
Load Accmtr	LDAA	86	2	2	96	3	2	A6	5	2	B6	4	3				M → A	•	•	↕	↕	R	•
	LDAB	C6	2	2	D6	3	2	E6	5	2	F6	4	3				M → B	•	•	↕	↕	R	•
Or, Inclusive	ORAA	8A	2	2	9A	3	2	AA	5	2	BA	4	3				A + M → A	•	•	↕	↕	R	•
	ORAB	CA	2	2	DA	3	2	EA	5	2	FA	4	3				B + M → B	•	•	↕	↕	R	•
Push Data	PSHA													36	4	1	A → M_{SP}, SP − 1 → SP	•	•	•	•	•	•
	PSHB													37	4	1	B → M_{SP}, SP − 1 → SP	•	•	•	•	•	•
Pull Data	PULA													32	4	1	SP + 1 → SP, M_{SP} → A	•	•	•	•	•	•
	PULB													33	4	1	SP + 1 → SP, M_{SP} → B	•	•	•	•	•	•
Rotate Left	ROL							69	7	2	79	6	3				M	•	•	↕	↕	⑥	↕
	ROLA													49	2	1	A	•	•	↕	↕	⑥	↕
	ROLB													59	2	1	B	•	•	↕	↕	⑥	↕
Rotate Right	ROR							66	7	2	76	6	3				M	•	•	↕	↕	⑥	↕
	RORA													46	2	1	A	•	•	↕	↕	⑥	↕
	RORB													56	2	1	B	•	•	↕	↕	⑥	↕
Shift Left, Arithmetic	ASL							68	7	2	78	6	3				M	•	•	↕	↕	⑥	↕
	ASLA													48	2	1	A	•	•	↕	↕	⑥	↕
	ASLB													58	2	1	B	•	•	↕	↕	⑥	↕
Shift Right, Arithmetic	ASR							67	7	2	77	6	3				M	•	•	↕	↕	⑥	↕
	ASRA													47	2	1	A	•	•	↕	↕	⑥	↕
	ASRB													57	2	1	B	•	•	↕	↕	⑥	↕
Shift Right, Logic	LSR							64	7	2	74	6	3				M	•	•	R	↕	⑥	↕
	LSRA													44	2	1	A	•	•	R	↕	⑥	↕
	LSRB													54	2	1	B	•	•	R	↕	⑥	↕
Store Accmtr	STAA				97	4	2	A7	5	2	B7	5	3				A → M	•	•	↕	↕	R	•
	STAB				D7	4	2	E7	6	2	F7	5	3				B → M	•	•	↕	↕	R	•
Subtract	SUBA	80	2	2	90	3	2	A0	5	2	B0	4	3				A − M → A	•	•	↕	↕	↕	↕
	SUBB	C0	2	2	D0	3	2	E0	5	2	F0	4	3				B − M → B	•	•	↕	↕	↕	↕
Subtract Accmtrs	SBA													10	2	1	A − B → A	•	•	↕	↕	↕	↕
Subtr with Carry	SBCA	82	2	2	92	3	2	A2	5	2	B2	4	3				A − M − C → A	•	•	↕	↕	↕	↕
	SBCB	C2	2	2	D2	3	2	E2	5	2	F2	4	3				B − M − C → B	•	•	↕	↕	↕	↕
Transfer Accmtrs	TAB													16	2	1	A → B	•	•	↕	↕	R	•
	TBA													17	2	1	B → A	•	•	↕	↕	R	•
Test, Zero or Minus	TST							6D	7	2	7D	6	3				M − 00	•	•	↕	↕	R	R
	TSTA													4D	2	1	A − 00	•	•	↕	↕	R	R
	TSTB													5D	2	1	B − 00	•	•	↕	↕	R	R
																		H	I	N	Z	V	C

Not: Durum göstericisindeki numaralarla verilmiş değişiklikler için daha sonraki tablolara bakınız.

Add (ADDA, ADDB) : Akümülatöre topla
Add Accumulators (ABA) : Akümülatörleri topla
Add with Carry (ADCA, ADCB) : Elde ile akümülatöre topla
Negate (NEG, NEGA, NEGB) : İkinin tümlevini al
Decimal Adjust A (DAA) : A akümülatörünü ondalık sayıya çevirir.
Subtract (SUBA, SUBB) : Akümülatörden çıkar
Subtract accumulators (SBA) : Akümülatörleri çıkar
Subtract with Carry (SBCA, SBCB) : Elde ile akümülatörden çıkar

işlemleridir. İşlevleri gerek adlarından gerekse Tablo 3.3'den anlaşılabilecek olan buişlemlerden DAA komutunu burada biraz daha açıklayalım. Bu komut İkili - Kodlanmış - Onlu (BCD) aritmetikte yalnız ABA, ADC ve ADD komutlarından sonra kullanılır. BCD olarak verilmiş iki sayının toplanmasında işlem, bu sayılar sanki normal ikili tabana göre verilmiş sayılar gibi yapıldığından, DAA komutu sonucun tekrar BCD bir sayıya dönüştürülmesini sağlar. BCD iki sayı ile çıkarma işleminde DAA komutu doğrudan doğruya kullanılamaz. Bu tür bir işlem için kısa da olsa özel bir program yazılması gerekir.

3.3.2. Lojik İşlemler:

Tablo 6.4'te gösterilen bu işlemler:

And (ANDA, ANDE) : VE
Complenement 1's (COM, COMA, COMB) : Birim tümlevini al
Exclusive OR (EORA, EOR B) : Dışlayıcı VEYA
Or inclusive (ORA, ORB) : VEYA

işlemleridir. Görülebileceği gibi tümlev alma işlevi akümülatörler için olduğu gibi bellek için de geçerlidir. AND ve OR komutları özellikle bir saklayıcının veya bellek konumunun belirli bitlerini "1" veya "0" duruma getirmekte kullanılır.

3.3.3. Veri sınama işlemleri:

Tablo 3.5'te özetlenen bu işlemler

Bit test (BITA, BITB) : Bit Sınama
Compare (CMPA, CMPB) : Karşılaştırma
Compare Accumulators (CBA) : Akümülatörleri karşılaştırma
Test, Zero or Minus (TST, TSTA, TSTB): Sıfır veya negatif mi sınama

İŞLEMLER	SİMGE	İŞLEVİ (Etiketler saklayıcı içeriklerini gösterir)	DURUM SAKLAYICISI 5 H	4 I	3 N	2 Z	1 V	0 C
Add	ADDA	A + M → A	↕	•	↕	↕	↕	↕
	ADDB	B + M → B	↕	•	↕	↕	↕	↕
Add Acmltrs	ABA	A + B → A	↕	•	↕	↕	↕	↕
Add with Carry	ADCA	A + M + C → A	↕	•	↕	↕	↕	↕
	ADCB	B + M + C → B	↕	•	↕	↕	↕	↕
Complement, 2's (Negate)	NEG	00 − M → M	•	•	↕	↕	①	②
	NEGA	00 − A → A	•	•	↕	↕	①	②
	NEGB	00 − B → B	•	•	↕	↕	①	②
Decimal Adjust, A	DAA	İkiliden BCD'ye	•	•	↕	↕	↕	③
Subtract	SUBA	A − M → A	•	•	↕	↕	↕	↕
	SUBB	B − M → B	•	•	↕	↕	↕	↕
Subract Acmltrs.	SBA	A − B → A	•	•	↕	↕	↕	↕
Subtr. with Carry	SBCA	A − M − C → A	•	•	↕	↕	↕	↕
	SBCB	B − M − C → B	•	•	↕	↕	↕	↕

Notlar: Bite bakılır, doğru ise "1", aksi takdirde "0" yapılır.

1- (Bit V) Test: Sonuç = 10000000?

2- (Bit C) Test: Sonuç = 00000000?

3- (Bit C) Test: BCD karekterin on tabanına göre değeri 9'dan büyük mü?
(Daha önce "1"se "0" yapılmaz)

Tablo 3.3. Aritmetik İşlemler

İŞLEM	SİMGE	İŞLEVİ	DURUM SAKLAYICISI 5 H	4 I	3 N	2 Z	1 V	0 C
And	ANDA	A • M → A	•	•	↕	↕	R	•
	ANDB	B • M → B	•	•	↕	↕	R	•
Complement, 1's	COM	$\overline{M} \rightarrow M$	•	•	↕	↕	R	S
	COMA	$\overline{A} \rightarrow A$	•	•	↕	↕	R	S
	COMB	$\overline{B} \rightarrow B$	•	•	↕	↕	R	S
Exclusive OR	EORA	A ⊕ M → A	•	•	↕	↕	R	•
	EORB	B ⊕ M → B	•	•	↕	↕	R	•
Or, Inclusive	ORA	A + M → A	•	•	↕	↕	R	•
	ORB	B + M → B	•	•	↕	↕	R	•

Tablo 3.4. Lojik İşlemler.

İŞLEM	SİMGE	İŞLEVİ	DURUM SAKLAYICISI 5 H	4 I	3 N	2 Z	1 V	0 C
Bit Test	BITA	A • M	•	•	↕	↕	R	•
	BITB	B • M	•	•	↕	↕	R	•
Compare	CMPA	A – M	•	•	↕	↕	↕	↕
	CMPB	B – M	•	•	↕	↕	↕	↕
Compare Acmltrs	CBA	A – B	•	•	↕	↕	↕	↕
Test, Zero or Minus	TST	M – 00	•	•	↕	↕	R	R
	TSTA	A – 00	•	•	↕	↕	R	R
	TSTB	B – 00	•	•	↕	↕	R	R

Tablo 3.5. Veri sınama işlemleri

işlemleridir. Bit sınama işlemi temelde bir VE işlemi olup bir verinin belirli bir bitinin "1" mi, "0" mı olduğunun anlaşılmasına olanak verir. Bu işlem için AND veya OR komutları da kullanılabilir. Fakat bu komutlar, Tablo 3.4'ten de görüleceği üzere akümülatörün içeriğini değiştirir.

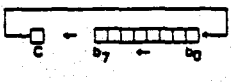
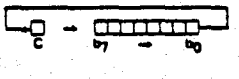
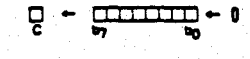
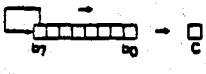
Karşılaştırma işlemi bir çıkarma işlemidir, fakat yine akümülatörlerden veya bellek konumundan, bunların içeriklerini değiştirmeden sıfır çıkarmaktır.

3.3.4. Veri üzerinde işlemler:

Veri üzerinde yapılacak işlemler için kullanılabilecek komutlar Tablo 3.6'da gösterilmiş olup aşağıda açıklanmıştır.

Clear (CLR, CLRA, CLRB) : Sıfırlama
Decrement (DEC, DECA, DECB) : Bir azalt
Increment (INC, INCA, INCB) : Bir arttır
Load Accumulator (LDAA, LDAB) : Akümülatörü yükle
Push Data (PSHA, PSHB) : Veri it (Akümülatörden yığına)
Pull Data (PULA, PULB) : Veri çek (Yığından akümülatöre)
Rotate Left (ROL, ROLA, ROLB) : Sola döndür
Rotate Right (ROR, RORA, RORB) : Sağa döndür
Shift Left, Arithmetic (ASL, ASLA, ASLB) : Sola kaydır (aritmetik)
Shift Right, Arithmetic (ASR, ASRA, ASRB): Sağa kaydır (aritmetik)
Shift Right, Logic (LSR, LSRA, LSRB) : Akümülatörleri sakla
Transfer Accumulators (TAB, TBA) : Akümülatörleri aralarında değiştir.

Tablo 3.6. Veri Üzerinde İşlemler

İŞLEMLER	SİMGE	İŞLEVİ (Etiketler saklayıcı içeriklerini gösterir)	DURUM SAKLAYICISI 5 H	4 I	3 N	2 Z	1 V	0 C
Clear	CLR	00 → M	•	•	R	S	R	R
	CLRA	00 → A	•	•	R	S	R	R
	CLRB	00 → B	•	•	R	S	R	R
Decrement	DEC	M − 1 → M	•	•	↕	↕	④	•
	DECA	A − 1 → A	•	•	↕	↕	④	•
	DECB	B − 1 → B	•	•	↕	↕	④	•
Increment	INC	M + 1 → M	•	•	↕	↕	⑤	•
	INCA	A + 1 → A	•	•	↕	↕	⑤	•
	INCB	B + 1 → B	•	•	↕	↕	⑤	•
Load Acmltr	LDAA	M → A	•	•	↕	↕	R	•
	LDAB	M → B	•	•	↕	↕	R	•
Push Data	PSHA	A → M_{SP}, SP − 1 → SP	•	•	•	•	•	•
	PSHB	B → M_{SP}, SP − 1 → SP	•	•	•	•	•	•
Pull Data	PULA	SP + 1 → SP, M_{SP} → A	•	•	•	•	•	•
	PULB	SP + 1 → SP, M_{SP} → B	•	•	•	•	•	•
Rotate Left	ROL	M	•	•	↕	↕	⑥	↕
	ROLA	A 	•	•	↕	↕	⑥	↕
	ROLB	B	•	•	↕	↕	⑥	↕
Rotate Right	ROR	M	•	•	↕	↕	⑥	↕
	RORA	A 	•	•	↕	↕	⑥	↕
	RORB	B	•	•	↕	↕	⑥	↕
Shift Left, Arithmetic	ASL	M	•	•	↕	↕	⑥	↕
	ASLA	A 	•	•	↕	↕	⑥	↕
	ASLB	B	•	•	↕	↕	⑥	↕
Shift Right, Arithmetic	ASR	M	•	•	↕	↕	⑥	↕
	ASRA	A 	•	•	↕	↕	⑥	↕
	ASRB	B	•	•	↕	↕	⑥	↕
Shift Right, Logic.	LSR	M	•	•	R	↕	⑥	↕
	LSRA	A 	•	•	R	↕	⑥	↕
	LSRB	B	•	•	R	↕	⑥	↕
Store Acmltr.	STAA	A → M	•	•	↕	↕	R	•
	STAB	B → M	•	•	↕	↕	R	•
Transfer Acmltrs	TAB	A → B	•	•	↕	↕	R	•
	TBA	B → A	•	•	↕	↕	R	•

Notlar: 4- (Bit V) Test: Operand=İşlemden önce 10000000 ?
5- (Bit V) Test: Operand=İşlemden önce 01111111 ?
6- (Bit V) Test: kaydırmadan sonra N⊕C'nin sonucuna eşit yapılır.

Yukarıdaki komutlardan CLR, DEC ve INC komutları bir akümülatörün veya bir bellek konumunun içeriklerini belirtilen şekilde değiştirmekte kullanılır. Bellek üzerinde yapılan işlemlerde sadece genişletilmiş ve indisli adresleme kullanılabilir. Yazılması basit olmasına rağmen yürütülmeleri için 6-7 makina çevirimi gerekir, çünkü verinin önce MİB'ne alınması, burada değiştirilmesi, sonra da tekrar bellekteki yerine yüklenmesi gerekir.

PSHA veya PSHB komutları A veya B akümülatörünün içeriğinin yığın göstericisinin işaret ettiği bellek konumuna kopya eder. Komut, SP'nin yine boş bir konumu gösterebilmesi için bir azalmasını da sağlar.

PULA veya PULB komutlarında ise yığın göstericisinin işaret etmekte olduğu konumdaki veri A veya B akümülatörüne alınır ve SP bir artır.

Döndürme ve kaydırma (ROTATE ve SHIFT) işlemlerinde Durum Saklayıcısının C (ELDE) flip-flopu da yer alır. Bir ASL işleminde verinin en önemli biti (MSB) veri sözcükten C flip-flopuna kaydırılır, diğer bitlerde 1 sola (örneğin b6, b7'ye b5, b6'ya) kayarlar. En önemsiz bit (LSB) bir "0" ile doldurulur. İşlem verinin 2 ile çarpılmasına eşdeğerdir.

ASR işlemi yukarıda açıklananın tersidir, fakat MSB'nin değeri korunarak yeni MSB olarak bırakılır. Böylece işaretli sayıların işareti korunmuş olur. Bu işlem de verinin iki ile bölünmesine eşdeğerdir.

LSR işleminde her bir sağa kayar, LSB C flip-flopuna alınır, MSB ise işlem sonunda "0" olur.

Döndürme (ROR) komutlarının işlevi Tablo 3.5'den görülebilmektedir. Kullanılabileceği bir çok yer vardır. Bir veri içerisindeki "1" veya "0" ların sayılması (yani PARİTE'nin tesbiti) bir örnek olarak verilebilir.

3.4. Program denetleme işlemleri:

Program denetleme işleri Tablo 3.7'de simgesel gösterimleri, adresleme türleri, makina kodları, gerektirdikleri bayt ve makina çevrim sayıları işlevleri ve Durum Saklayıcısı üzerindeki etkileri ile birlikte verilmiştir. Bu tablodan görülebileceği ü üzere program denetleme işlemleri kendi aralarında (1) İmdis Saklayıcısı/Yığın Göstericisi komutları (2) Atlama ve Dallanma komutları olarak iki gruba ayrılabilir.

3.4.1. İndis Saklayıcısı/Yığın Göstericisi Komutları:

Bu komutlar

Compare Indek Register (CPX) :İndis Saklayıcısını karşılaştır
Decrement Index Register (DEX) :İndis Saklayıcısını bir azalt
Decrement Stack Pointer (DES) : Yığın Göstericisini bir azalt
Increment Index Register (INX) :İmdis Saklayıcısını bir artır
Incerement Stack Pointer (INS) : Yığın Göstericisini bir artır
Load Index Register (LDX) :İndis Saklayıcısını yükle
Load Stack Pointer (LDS) : Yığın Göstericisini yükle
Store Index Register (STX) :Indis Saklayıcısını sakla
Store Stack Pointer (STS) : Yığın Göstericisini sakla
Index Register-Stack Pointer (TXS): Indis Saklayıcısını Yığın Göstericisine aktar
Stack Pointer-Index Register(TSX): Yığın Göstericisini İndis Saklayıcısına aktar.

komutlarıdır. Bunlardan ilk dokuzunun işlevleri komut adlarından da anlaşılabilir.

Bir mikroişlemcinin ilk çalışmaya başlatılmasında mutlaka yapılması gereken bir işlem RAM bellekte bir bölümü yığın için ayırmak ve buranın en büyük adresli konumunu Yığın Göstericisine yüklemektir. Örneğin yığın için 0400 ve 04FF arası ayrılırsa mikroişlemciye verilecek ilk komutlardan birinin LDS # 04FF olması gerekir (initialization).

TSK ve TXS komutları oldukça özel komutlardır. TSX komutu, İndis Saklayıcının, yığına itilmiş olan son verinin adresi ile yüklenmesini sağlar. TXS komutu ise Yığın Göstericisinin, İndis Saklayıcının o andaki değerinden bir eksiği ile yüklenmesi sonucunu doğurur. Dolayısıyla da bir TXS komutundan sonra PULL komutu kullanılırsa yığından alınan veri İndis Saklayıcısının gösterdiği yerdeki veri olur.

3.4.2. Atlama ve dallanma komutları:

Tablo 3.7.'de özetlenmiş olan bu komutlar

Branch Always (BRA) : Daima dallan
Branch if Carry Clear (BCC) : Elde yoksa (C = 0) dallan
Branch if Carry Set (BCS) : Elde varsa (C = 1) dallan
Branch if Equal to Zero (BEQ) : Sonuç sıfırsa (Z = 1) dallan
Branch if Greater than or Equal to Zero (BGE): Sonuç sıfır veya daha büyükse dallan
Branch if Greater Than Zero (BGT): Sonuç sıfırdan büyükse dallan
Branch if Higher (BHI) : Daha büyükse dallan

POINTER OPERATIONS	MNEMONIC	IMMED OP	IMMED ~	IMMED #	DIRECT OP	DIRECT ~	DIRECT #	INDEX OP	INDEX ~	INDEX #	EXTND OP	EXTND ~	EXTND #	IMPLIED OP	IMPLIED ~	IMPLIED #	BOOLEAN/ARITHMETIC OPERATION	COND. CODE REG. 5 H	4 I	3 N	2 Z	1 V	0 C
Compare Index Reg	CPX	8C	3	3	9C	4	2	AC	6	2	BC	5	3				$X_H - M, X_L - (M+1)$	•	•	①	•	②	•
Decrement Index Reg	DEX													09	4	1	$X - 1 \rightarrow X$	•	•	•	↕	•	•
Decrement Stack Pntr	DES													34	4	1	$SP - 1 \rightarrow SP$	•	•	•	•	•	•
Increment Index Reg	INX													08	4	1	$X + 1 \rightarrow X$	•	•	•	↕	•	•
Increment Stack Pntr	INS													31	4	1	$SP + 1 \rightarrow SP$	•	•	•	•	•	•
Load Index Reg	LDX	CE	3	3	DE	4	2	EE	6	2	FE	5	3				$M \rightarrow X_H, (M+1) \rightarrow X_L$	•	•	↕	↕	R	•
Load Stack Pntr	LDS	8E	3	3	9E	4	2	AE	6	2	BE	5	3				$M \rightarrow SP_H, (M+1) \rightarrow SP_L$	•	•	↕	↕	R	•
Store Index Reg	STX				DF	5	2	EF	7	2	FF	6	3				$X_H \rightarrow M, X_L \rightarrow (M+1)$	•	•	↕	↕	R	•
Store Stack Pntr	STS				9F	5	2	AF	7	2	BF	6	3				$SP_H \rightarrow M, SP_L \rightarrow (M+1)$	•	•	↕	↕	R	•
Indx Reg → Stack Pntr	TXS													35	4	1	$X - 1 \rightarrow SP$	•	•	•	•	•	•
Stack Pntr → Indx Reg	TSX													30	4	1	$SP + 1 \rightarrow X$	•	•	•	•	•	•

[Motorola, 8 Bit Microprocessor data book]

A. İndis ve yığın göstericileri üzerinde işlemler

OPERATIONS	MNEMONIC	RELATIVE OP	RELATIVE ~	RELATIVE #	INDEX OP	INDEX ~	INDEX #	EXTND OP	EXTND ~	EXTND #	IMPLIED OP	IMPLIED ~	IMPLIED #	BRANCH TEST	COND. CODE REG. 5 H	4 I	3 N	2 Z	1 V	0 C
Branch Always	BRA	20	4	2										None	•	•	•	•	•	•
Branch If Carry Clear	BCC	24	4	2										C = 0	•	•	•	•	•	•
Branch If Carry Set	BCS	25	4	2										C = 1	•	•	•	•	•	•
Branch If = Zero	BEQ	27	4	2										Z = 1	•	•	•	•	•	•
Branch If ≥ Zero	BGE	2C	4	2										$N \oplus V = 0$	•	•	•	•	•	•
Branch If > Zero	BGT	2E	4	2										$Z + (N \oplus V) = 0$	•	•	•	•	•	•
Branch If Higher	BHI	22	4	2										C + Z = 0	•	•	•	•	•	•
Branch If ≤ Zero	BLE	2F	4	2										$Z + (N \oplus V) = 1$	•	•	•	•	•	•
Branch If Lower Or Same	BLS	23	4	2										C + Z = 1	•	•	•	•	•	•
Branch If < Zero	BLT	2D	4	2										$N \oplus V = 1$	•	•	•	•	•	•
Branch If Minus	BMI	2B	4	2										N = 1	•	•	•	•	•	•
Branch If Not Equal Zero	BNE	26	4	2										Z = 0	•	•	•	•	•	•
Branch If Overflow Clear	BVC	28	4	2										V = 0	•	•	•	•	•	•
Branch If Overflow Set	BVS	29	4	2										V = 1	•	•	•	•	•	•
Branch If Plus	BPL	2A	4	2										N = 0	•	•	•	•	•	•
Branch To Subroutine	BSR	8D	8	2											•	•	•	•	•	•
Jump	JMP				6E	4	2	7E	3	3				See Special Operations	•	•	•	•	•	•
Jump To Subroutine	JSR				AD	8	2	BD	9	3					•	•	•	•	•	•
No Operation	NOP										01	2	1	Advances Prog. Cntr. Only	•	•	•	•	•	•
Return From Interrupt	RTI					↕					3B	10	1		①					
Return From Subroutine	RTS										39	5	1		•	•	•	•	•	•
Software Interrupt	SWI										3F	12	1	See Special Operations	•	•	•	•	•	•
Wait for Interrupt*	WAI										3E	9	1		•	②	•	•	•	•

*WAI puts Address Bus, R/W, and Data Bus in the three-state mode while VMA is held low.

[Motorola, 8 Bit Microprocessor data book]

B. Atlama ve dallanma işlemleri

Notlar: 7- (Bit N) Test: Sonucun en önemli baytının işareti=1?

8- (Bit V) Test: En önemsiz baytların çıkarılmasından 2'nin tümlevi taşma var mı?

9- (Bit N) Test: Sonuç sıfırdan küçükmü? (Bit 15-1)

10- (Tüm bitler) Yığından eski durum yüklenir.

11- (Bit I) Bir kesme gelirse 1 yapılır. Daha önce 1 ise BEKLEME durumuna geçmek için bir NMI gelmesi gerekir.

Tablo 3.7. Program denetleme işlemleri.

Branch if Less than or Equal to Zero (BLE): Sıfır veya daha küçükse dallan
Branch if Greater Than Zero (BGT) : Sonuç sıfırdan büyükse dallan
Branch if Higher (BHI) : Daha büyükse dallan
Branch if less than or Equal to Zero (BLE): Sıfır veya daha küçükdallan
Branch if Lower or Same (BLS) : Daha küçükse veya aynı ise dallan
Branch if Minus (BIM) : Negatifse (N 1) dallan
Branch if not Equal to Zero (BNE): Sıfıra eşit değilse (Z = 0) dallan
Branch if Overflow Clear (BVC) : Taşma yoksa (V = 0) dallan
Branch to Subroutine (BSR) : Altyordama dallan
Jump (JMP) : Atla
Jump to Subroutine (JSR) : Altyordama atla
No Operation (NOP) : İşlem yapma
Return from Interrupt (RII) : Kesme yordamından dön
Return from Subroutine (RTS) : Altyordamdan dön
Software Interrupt (SWI) : Yazılım kesmesi
Wait for Interrupt (WAI) : Kesme bekle

komutlarıdır. Bu komutlardan en basiti JMP komutu olup Program Sayıcısının yeni bir değerle yüklenmesine ve böylece programın yeni bir konuma atlamasına neden olur.

Diğer bir atlama komutu BRA komutudur. Ayrıt 3.2.5'de de değinildiği gibi tüm "BRANÇH" komutları bağıl adresleme kullanan iki baytlık komutlardır. BRA komutunda atlanılacak (dallanılacak) konum PC'ye öteleme verisinin eklenmesi ile bulunacağından atlanabilecek konum program Sayıcısına göre +129 veya -126 baytlık bir alan içerisinde yer alır.

BRA komutu koşulsuz bir dallanma komutu olup bit atlama komutuna eşdeğerdir. Buna karşın BCS, BCC, BEQ, BNE, BMI, BPL, BVS ve BVC gibi komutlarda dallanma belirli bir koşulun var olup olmadığına bağlıdır. Koşullu dallanma komutları Durum Saklayıcısının belirli bitlerinin veya birleşimlerinin "1" veya "0" olup olmadığına bağlı olarak bir dallanma sağlar. Koşul gerçekleşmişse Program Sayıcısı değiştirilmeyerek bulunulan konumdan programa devam edilir. En fazla kullanılanları C, Z, N ve V bitlerini sınayan 4 çift komut olup aşağıda sıralanmıştır.

Komut	Simge	Op kodu	Koşul
Elde varsa dallan	BCS	25	C = 1
Elde yoksa dallan	BCC	24	C = 0
Sıfıra eşitse dallan	BEQ	27	Z = 1
Sıfıra eşit değilse dallan	BNE	26	Z = 0
Negatifse dallan	BMI	2B	N = 1
Pozitifse dallan	BPL	2A	N = 0
Taşma varsa dallan	BVS	29	V = 1
Taşma yoksa dallan	BVC	28	V = 0

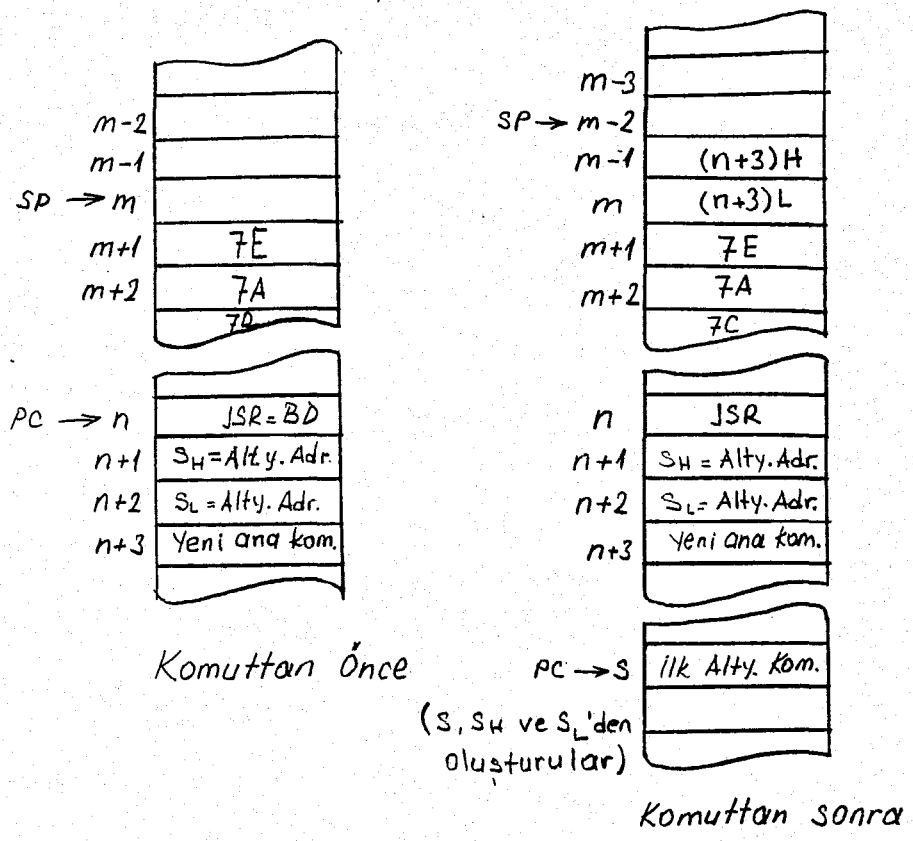
Diğer koşulla dallanma komut çiftlerinden BHI (Daha büyükse dallan) ve BLS (Daha küçük veya aynı ise dallan) komutlara bir anlamda BCS ve BCC komutlarına benzerler fakat sadece "C" bitini değil "Z" bitini de sınarlar. Bir CMP (karşılaştırma) veya SUB (çıkarma) komutundan sonra kullanılırsa BHI, akümülatördeki değer operanddan büyükse dallanmaya neden olur, BLS ise akümülatördeki işaretsiz sayı operanda eşit veya daha küçükse bir dallanma sağlar.

Koşullu dallanma komutlarında geriye kalan BLT ve BGE ile BLE ve BGT çiftleri ise özellikle işaretli sayılar üzerinde yapılan işlem sonuçlarını sınamakta kullanılırlar. Durum Saklayıcısının bitleri üzerimde yaptıkları sınama, Tablo 3.7'de verilmiştir.

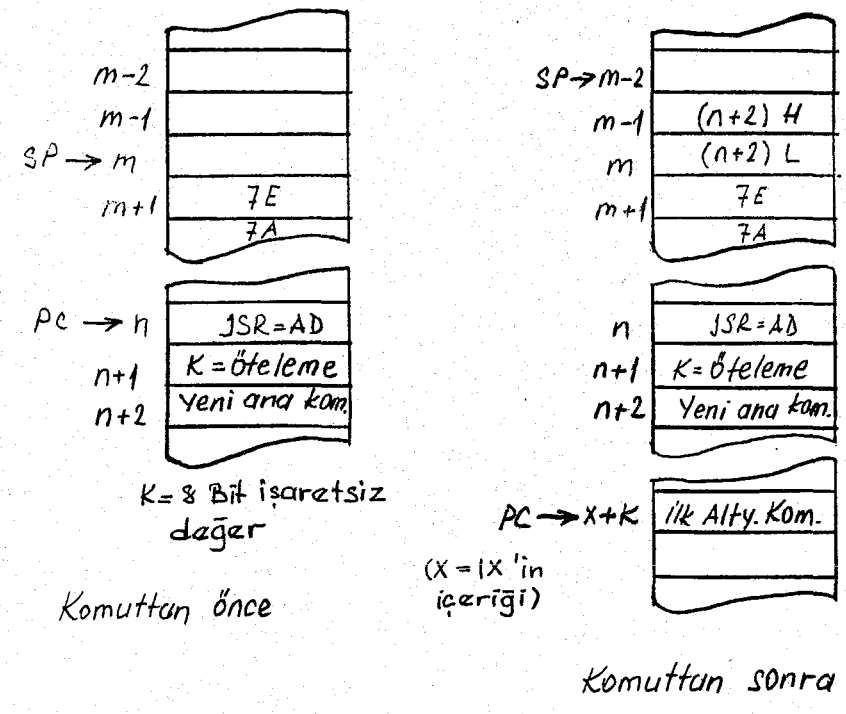
Program denetleme komutlarından JSR ve BSR komutları program içerisinde bir altyordam çağrılmasına olanak verirler. BSR komutunun kullanılabilmesi için altyordam adresinin bulunulan konumdan en fazla +129 bayt ötede veya -126 bayt geride olması gerekir. Altyordam yürütüldükten sonra tekrar ana programa dönülmesi için program dönüş adresi yığında saklanır.

JSR ve BSR komutlarının yürütülmüş şekli Şekil 3.4'de özetlenmiştir. Örneğin genişletilmiş adreslemeli JSR komutunda (makina kodu:BD) yapılan işlemler şunlardır.

1. Bir sonraki komut adresinin alt baytı Yığın Göstericisinin işaret ettiği konuma yüklenir.
2. Yığın Göstericisi bir azaltılır.
3. Bir sonraki komut adresinin üst baytı Yığın Göstericisinin işaret ettiği yeni konuma yazılır.
4. Yığın Göstericisi yine bir azaltılır.
5. Program Sayıcısı JSR komutunun bir parçası olan altyordam başlangıçta adresi ile yüklenir.

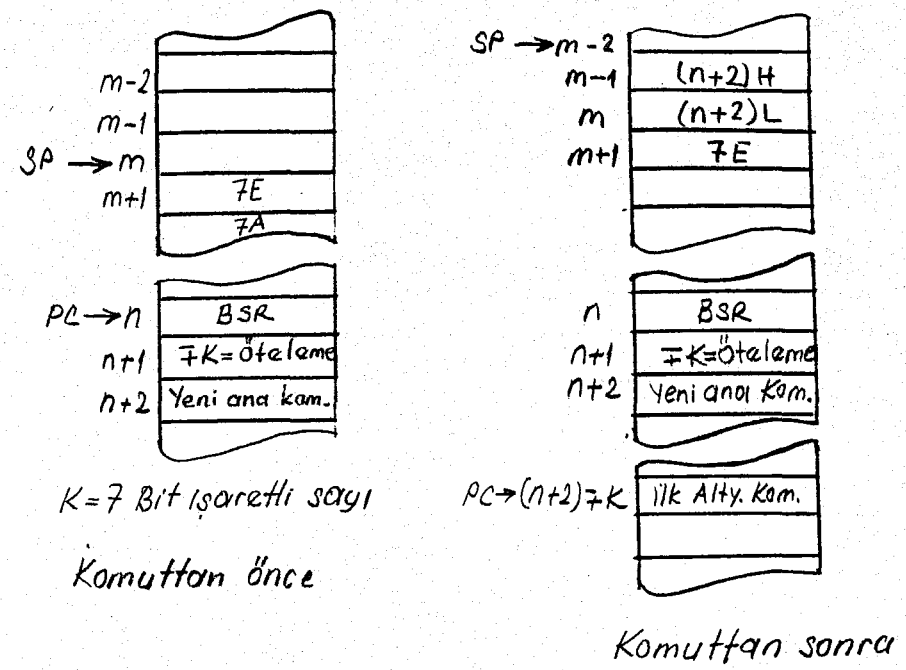


a. Genişletilmiş adreslemeli JSR komutunda



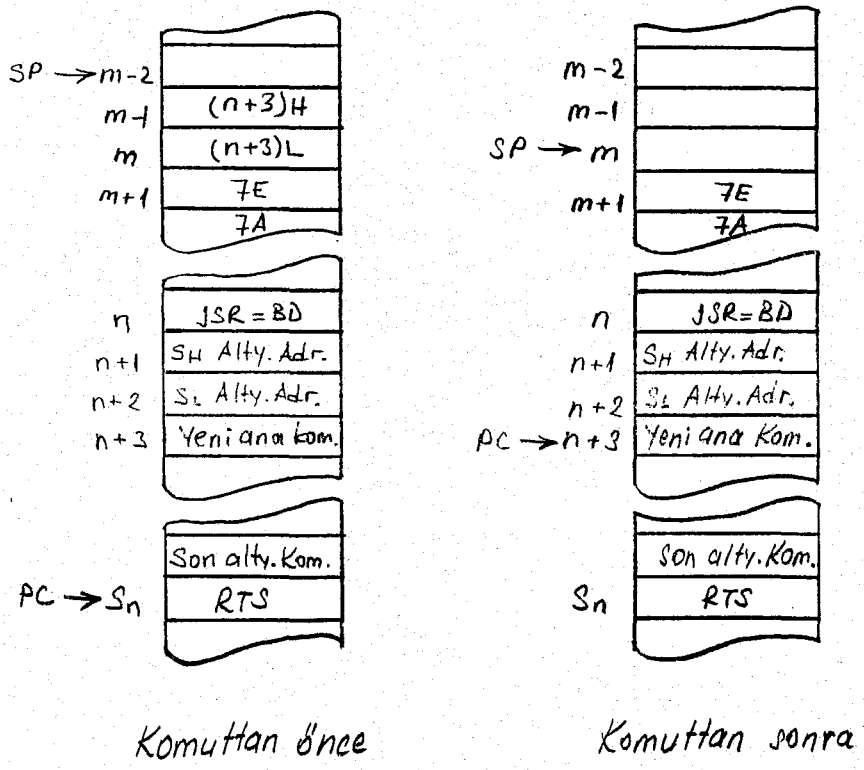
b. İndisli adreslemeli JSR komutunda

Şekil 3.4. JSR ve BSR komutlarında program akışı



c. BSR komutunda

Şekil 3.4. (Devam)



Şekil 3.5. RTS komutunda program akışı

Böylece program denetimi altyordama aktarılmış olur.

Alt yordandan ana programa dönüş RTS (Return from Subroutine-Altyordamdan dön) komutu ile gerçekleştirilir. Her altyordamın sonunda bu komut mutlaka bulunmalıdır. Bu komut üzerine, Şekil 3.5'te gösterildiği gibi,

1. Yığın Göstericisi bir arttırılır.

2. Yığın Göstericisinin işaret ettiği konumdaki bayt program sayıcısının üst baytı olarak içeri alınır.

3. Yığın göstericisi bir artırılır.

4. Yığın Göstericisinin şimdi işaret ettiği konumdaki bayt program Sayıcısının üst baytına yazılır.

Böylece ana programa dönülmüş olur.

Tablo 3.7'da program denetleme komutları arasında gösterilen NOP (No Operation-İşlem Yapma) komutu çok sınırlı bir anlamda bir atlama komutudur. Tek etkisi Program Sayıcısını bir arttırmaktır. Program geliştirme sırasında, program içine sonradan bazı komutlar eklenmesi gerekebileceği düşüncesi ile, bazı yerlere bu NOP komutu koyulur. Programın çeşitli dallardan akış süresini eşitlemek için kullanılabilir.

WAI (Wait for Interrup-Kesme bekle) komutu mikroişlemcinin kesmeli çalışması gereken durumlarda bir kesme istemine cevap süresini kısaltmak için kullanılır. WAI komutu, bir kesme istemi üzerine yığına atılması gereken saklayıcıları Ayrıt 4.1'de açıklanan şekilde yığına atar ve kesmeyi bekler. Böylece kesme geldiği an, bu saklayıcılar halihazırda yığında olduğundan, kesme hizmet yordamına atlanması çok daha kısa bir sürede (14 makina çevrimi yerine 6 makina çevriminde) gerçekleşir.

SWI (Software Interrupt-Yazılım kesmesi) komutuna daha önce Ayrıt 4.1'de değinilmişti. Bu komut birçok değişik amaçla kullanılabilir. Örneğin program geliştirme sırasında programın akışının denetlenmesinde kullanılabilir. Programa yerleştirilen bir SWI komutu bütün saklayıcıları yığına iteceğinden o andaki durum buradan okunabilir.

3.5. Durum Göstericisi İşlemleri:

Durum Saklayıcısının bitlerini değiştirebilmemizi sağlayan komutlar Tablo 3.8'de özetlenmiştir. Görülebileceği üzere:

1. C, V ve I bitlerini "1" veya "0" yapmamızı sağlayan komutlar vardır. Böylece örneğin kesme denetimi yapılabilir.

2. Durum Saklayıcısının belirli bir andaki durumunu daha sonra kullanmak üzere saklamamız gerekirse TPA (Transfer from processor Condition Codes Register to Accumulator A - Durum

İŞLEM	SİMGE	IMPLIED OP	IMPLIED ~	IMPLIED #	İŞLEVİ	DURUM SAKLAYICISI 5 H	4 I	3 N	2 Z	1 V	0 C
Clear Carry	CLC	0C	2	1	0→C	•	•	•	•	•	R
Clear Interrupt Mask	CLI	0E	2	1	0→I	•	R	•	•	•	•
Clear Overflow	CLV	0A	2	1	0→V	•	•	•	•	R	•
Set Carry	SEC	0D	2	1	1→C	•	•	•	•	•	S
Set Interrupt Mask	SEI	0F	2	1	1→I	•	S	•	•	•	•
Set Overflow	SEV	0B	2	1	1→V	•	•	•	•	S	•
Acmltr A→CCR	TAP	06	2	1	A→CCR	———	———	———	①	———	———
CCR→Acmltr A	TPA	07	2	1	CCR→A	•	•	•	•	•	•

Not: 1- (Bütün bitler) Akümülatör Durum Saklayıcısına aktarılır.

Tablo 3.8. Durum Saklayıcısı İşlemleri

Saklayıcısını A Akümülatörüne aktar) komutunu kullanabiliriz. A Akümülatöründen de belleğe yükleme yapılabilir.

3. Yukarıda açıklanan işlemin tersi TAP (Transfer from Accumulator A to Processor Condition Code Register - A Akümülatörünü Durum Saklayıcısına aktar) komutu ile yapılabilir.

3.6. Programlama Örnekleri:

Bu bölümde açıklanmış bulunan komutların bir program içerisinde nasıl kullanılabileceğini gösterebilmek amacı ile aşağıda üç örnek program verilmiştir.

3.6.1. Bir veri dizisinin bir bellek alanından diğer bir alana aktarılması:

Başlangıç adresi $0001 ve $0002 adreslerinde, uzunluğu ise $0000 adresinde olan bir veri dizisini (uzunluğu 256'dan küçük olduğu varsayılmıştır) başlangıç adresi $0003 ve $0004 adreslerinde gösterilen bellek bölümüne aktarmamız gerektiğini düşünelim. Bu işlemi gerçekleştirebilmek için kullanılabilecek bir program aşağıda verilmiştir.

```
ETİKET    KOMUT    OPERAND      YORUM
BAŞLA     LDAA     UZUNLUK      Uzunluğu A'ya yükle
          STAA     SAYAÇ        A'yı sayaca at
DÖNGÜ     LDX      KAYNAK       Kaynak adresini IX'e al
          LDAA     0,X          İlk veriyi A'ya al
          INX                   Gelecek döngü için kaynak
          STX      KAYNAK       göstericisini bir arttır
          LDX      HEDEF        Hedef adresini IX'e al
          STAA     0,X          İlk veriyi yükle
```

ETİKET	KOMUT	OPERAND	YORUM
	INX		Gelecek döngü için hedef
	STX	HEDEF	göstericisini bir arttır
	DEC	SAYAÇ	Veri dizisi sayısını bir azalt
	BNE	DÖNGÜ	Daha veri varsa dön
	SWI		

Yukarıdaki programda operand alanında semboller kullanılmıştır. Örneğin KAYNAK sembolü (Ø0001) ve (Ø0002) dir, UZUNLUK ise (Ø0000) dır. Programı S0010 adresinden başlatırsak ve ØAYAÇ içinde Ø0050 adresini kullanırsak programın makina diline çevrilmiş şekli aşağıda gibi olur.

Adres (Hex)	Makina Kodu (Hex)	Etiket	Komut	Operand
0010	96 00	BAŞLA	LDA A	Ø 00
0012	97 50		STA A	Ø 50
0014	DE 01	DÖNGÜ	LDX	Ø 01
0016	A6 00		LDA A	X
0018	08		INX	
0019	DF 01		STX	Ø 01
001B	DE 03		LDX	Ø 03
001D	A7 00		STA A	X
001F	08		INC X	
0020	DF 03		STX	Ø 03
0022	7A 00 50		DEC	Ø 50
0025	26 ED		BNE	DÖNGÜ
0027	3F		SWI	

3.6.2. Başvuru tabloları:

Mikroişlemcilerin endüstriyel uygulamalarında başvuru tabloları sık sık kullanılır. Örneğin üç fazlı tam denetimli bir köprü doğrultucusunda fazların durumlarına göre hangi tristörlerin tetikleneceği için bir başvuru tablosu kullanılabilir. Veya bir dizi girişin (basınç, sıcaklık v.s.) durumuna göre açılması veya kapanması gereken vanalar yine bir başvuru tablosu yardımıyla bulunabilir. Daha somut bir örnek ise açıların sinüsünün bulunması olabilir. Aşağıda verilen program örneği üç bitlik bir sayının karesini bir başvuru tablosuna bakarak bulur. Örnekte karesi bulunacak sayının Ø0040 adresinde olduğu, programın Ø0000'dan, kareler tablosunun ise Ø0011'den başladığı varsayılmıştır.

Adres	Makina Kodu	Komut	Operand	Yorum
0000	86 00	LDA A	TABLOH	Tablonun üst baytını A'ya al
0002	97 0F	STA A	GEÇİCİ	Sakla
0004	96 40	LDA A	⌀ 40	Ötelemeyi al
0006	97 10	STA A	GEÇİCİ+1	Sakla
0008	DE 0F	LDX	GEÇİCİ	Ötelenmiş adresi IX'e al
000A	A6 11	LDA A	TABLO,X	Tablodan kareyi bul
000C	97 41	STA A	⌀ 41	Sonucu sakla
000E	3F	SWI		
000F	XX			
0010	XX			
0011	00			0'ın karesi
0012	01			1'in karesi
0013	04			2'nin karesi
0014	09			3'ün karesi
0015	10			4'ün karesi
0016	19			5'in karesi
0017	24			6'nın karesi
0018	31			7'nin karesi

Yukarıdaki programı açıklamak için ⌀40'da 04 sayısının olduğunu varsayalım. Dördüncü komuttan sonra ⌀000F ve ⌀0010 adreslerinde ⌀0004 bilgisi olacaktır. Beşinci komut bu verinin İndis Saklayıcısına yüklenmesini sağlar. Altıncı komut, İndis Saklayıcısının içeriğine 11 eklenerek elde edilen adresteki verinin Akümülatör A'ya yüklenmesini sağlar. Böylece bu akümülatörün içeriği ⌀0010 adresinin içeriği olan ⌀10 sayısı (4'ün karesinin heksadesimal eşdeğeri) olur. Yedinci komut bu sayıyı ⌀0041 adresine yükler.

3.6.3 Çarpma:

İkili tabanda çarpma yapmanın en kolay yolu çarpılanı çarpan kadar kendisiyle toplamaktır. Fakat bu çok uzun bir zaman alır. Çarpan sayıda sağdan sola doğru gidildikçe bit değerlerinin iki kat arttığı düşünülecek olursa çarpma işleminin sonucunu çok daha kısa bir sürede bulabilecek bir algoritma gerçekleştirebilir. Aşağıdaki algoritmada çarpan ve çarpılan sayıların işaretsiz olduğu varsayılmıştır.

Adım 1: Çarpımı sıfıra eşitle
Adım 2: Çarpanın en sağdaki bitine bak, sıfırsa Adım 3'e atla, birse çarpıma ekle
Adım 3: Çarpılan sayıyı en sağdaki bite 0 gelecek şekilde bir bayt sola kaydır, yani iki ile çarp.
Adım 4: Çarpan sayıyı sağa bir bit kaydır.
Adım 5: Eğer çarpan sayının bütün bitleri kullanılmışsa dur, kullanılmamışsa Adım 2'ye dön.

```
   45                 00101101              00101101
x  23               x 00010111              00010111
-----              ----------             ----------
  135                00101101              00101101
   90               00101101              00101101
-----              00101101              00101101
 1035              00000000              ----------          1035
                  00101101               010000001011        10
                 00000000
                00000000
               00000000
              ----------
               0000100000001011   1035
                                    10
```

a. b. c.

Şekil 3.6. Sekizer bitlik iki sayının çarpımı
a. Ondalık tabanda işlem
b. İkili tabanda işlem
c. Algoritma kullanarak işlem

Algoritma Şekil 3.6'da sekizer bitlik iki sayı için (45 ve 23) açıklanmıştır.

Yukarıda açıklanan algoritmanın bir mikroişlemci ile gerçekleştirilmesinde daha kolay bir yöntem çarpılanı sola kaydırıp çarpıma eklemek yerine çarpılanı sabit tutup çarpımı sağa kaydırmaktır. Bu durumda algoritma

Adım 1: Çarpımı sıfıra eşitle
Adım 2: Çarpanın en sağdaki bitine bak, sıfırsa Adım 3'e atla, birse çarpılanı çarpımın üst baytına ekle.
Adım 3: Çarpımı bir bit sağa kaydır.

Adım 4: Eğer çarpan sayının bütün bitlerine bakılmışsa dur, değilse Adım 2'ye dön.

şeklini alır. Bu algoritma için program aşağıda verilmiş olup çarpılanın $0080, çarpanın $0081 adreslerinde olduğu, sonucun ise $0082 (üst bayt) ve $0083 adreslerine yerleştirileceği varsayılmıştır. $0084 adresi çarpanın bit sayısını saymak için kullanılmıştır.

Adres (Hex)	Makina Kodu	Komut	Operand
0020	C6 08	LDA B	#8
0022	D7 84	STA B	$ 84
0024	D6 80	LDA B	$ 80
0026	4F	CLR A	
0027	74 00 84	LSR	$ 84
002A	24 01	BCC	$ 2D
002C	1B	ABA	
002D	46	ROR A	
002E	76 00 83	ROR	$ 83
0031	7A 00 84	DEC	$ 84
0034	26 F1	BNE	$ 27
0036	97 82	STA A	$ 82
0038	3F	SWI	

Yukarıdaki programda çarpımın üst baytı A akümülütöründe alt baytı ise $0083 adresinde tutulmuş sonuç elde edildiği zaman A akümülatörü $0082 adresine yüklenmiştir.

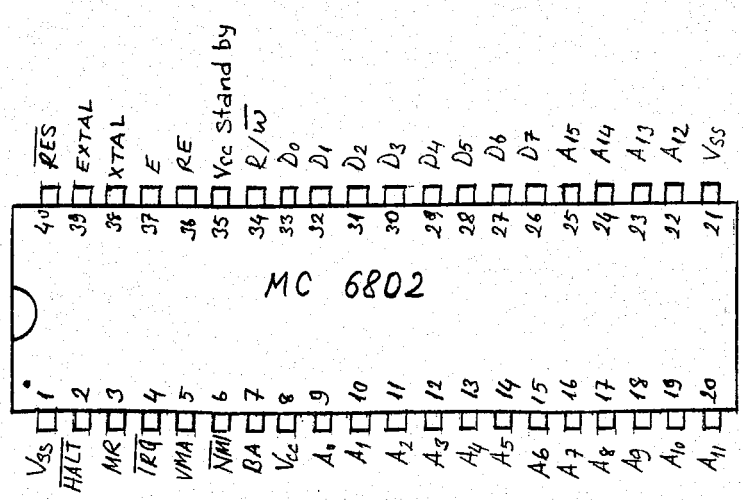
BÖLÜM-4
6802 MİKROİŞLEMCİLİ BİR MİKROBİLGİSAYAR DONANIMI (HARDWARE)

4.1. 6802 Mikroişlemci birimi

MC 6802 mikroişlemcisi 6800 ailesinin merkezi denetim işlemlerini yerine getiren 8-bitlik NMOS (N. channel metal oxide semiconductor) monolitik bir mikroişlemcidir. Yalnız 5V'luk besleme gerektirir. Bu özellik 6800 ailesi mikroişlemcileri için bir avantajdır.

6802,

- 16 Bitlik, adres taşıtı
- 72 tür komut
- 7 tür adresleme (Hemen, Direk, Genişletilmiş, İndisli, İçerilmiş, bağıl akümülatör adreslemeleri)
- Yığın
- Vektörlü sıfırlama
- Maskelenebilen kesme
- 8 bitlik iki akümülatör, indis saklayıcısı, program sayacı, yığın gösterici, durum saklayıcısı.
- 2.0 MHz'e kadar saat ($1/2 \times 10^{-6}$ sn. temel işlem zamanı)
- Durdurma (halt) ve tek komut yürütme
- Saat yongası içermesi
- 128 x 8 Bit RAM bellek. Adresi $0000 - $007F
- Stand by RAM bellek besleme ayrık ucu.



Şekil. 4.1. 6802 bacakları.
[Motorala, 8 Bit Microprocessor data book]

6802 mikroişlemcisi bacakları

Saat girişleri (EXTAL ve XTAL): Mikroişlemcisinin ve çevre birimlerinin çalışması bu saatle eşzamanlıdır. Saat'in birbirini izleyen iki yükselen kenarı arasındaki süreye bir makina çevrimi (machine cycle) adı verilir. Komutların yürütme süresi daima bu sürenin tam katıdır. En kısa komut 2 çevrimde, en uzun ise 12 çevrimde tamamlanır.

Adres Taşıtı (A0 - A15): Adres taşıtı için 16 ayak ayrılmıştır. Çıkışları, "üç-durumlu taşıt sürücüleri" (three-state bus driver) olup 1 standart TTL yükü (0.4V'ta 1,6 mA olarak tanımlanır) sürebilecek kapasitedir.

Veri Taşıtı (D0-D7): Veri taşıtı için 8 ayak ayrılmıştır. Taşıtı, çevre birimlerinden veya birimlerine veri transferini sağlayacak şekilde çift yönlüdür. Bu taşıtın da üç-durumlu çıkış tamponları olup yine bir standart TTL yükü sürebilecek kapasitededir.

BA (Bus available-Taşıt kullanıma hazır): Bu çıkış, normal durumda "0" düzeyindedir. "1" durumunda olduğu zaman, ki bu ancak HALT girişinin "0" olması veya mikroişlemcinin bir WAIT komutu yürütmüş olması dolayısıyla "bekleme" durumuna geçmiş olması ile olasıdır, adres ve veri taşıtları ile R/$\overline{W}$ denetim çıkışının 3. duruma çekilmiş olduğu anlamını verir.

R/$\overline{W}$ (Read / $\overline{write}$ - oku / $\overline{yaz}$): Bu çıkış, bellek ve diğer çıkış birimlerine mikroişlemcilerin okuma (R/$\overline{W}$ = 1) veya yazma (R/$\overline{W}$ = 0) durumunda olduğunu belirtir. Mikroişlemci bekleme (HALT) durumuna geçtiği zamanda, yüksek empedans durumunda olur.

$\overline{RESET}$ (sıfırlama): Bu giriş bir elektrik kesintisinden sonra veya ilk güç verilişinde mikroişlemcinin kendisini sıfırlaması ve yeniden çalışmaya başlamasını sağlar. Mikroişlemci çalışır durumda iken de, yeniden başlatma (reinitialize) için kullanılabilir. Eğer RESET girişi; minimum 8 saat peryodluk bir süreyle "0" düzeyinde tutulduktan sonra "1" düzeyine çıkarsa;

-Kesme maskesi (Durum saklayıcısının I biti) "1" durumuna geçer ve böylece kesmeler engellenir.

- VMA "0" düzeyime geçer.
- Veri taşıtı, yüksek empedans durumuna geçer.
- BA "0" olur.
- R/$\overline{W}$ "1" olur.
- Adres taşıtında $FFFE yer alır.

Böylece FFFE ve FFFF bellek konumlarında bilgi, (ki reset Vektör sıfırlama vektörü olarak bilinir) program sayıcısına yüklenir. Sıfırlama vektörü, programın başlaması gereken bellek gözünü gösterdiğinden, mikroişlemci ilk olarak bu bellek konumdaki komutu içeri alır. Mikroişlemcinin kesmelere cevap vermesi isteniyorsa, ilk yapılacak işlemlerden biri, kesme bitinin (I biti) "0" durumuna getirilmesi işlemi olması gerekir.

$\overline{\text{RESET}}$ hattı, böyle bir girişi olan ve ilk çalışma durumuna getirilmesi gereken diğer dış birimlere de, örneğin 6821 gibi çevre arabirimine de, bağlanabilir. Böylece mikroişlemci ile birlikte bu birimlerin da ilk çalışma durumuna girmesi sağlanır.

$\overline{\text{IRQ}}$ (Interrupt request-Kesme istemi): Düzeye duyarlı olan bu giriş, tipik olarak çevre birimlerinin mikroişlemci ile iletişim sağlamasında kullanılır. Eğer bu giriş "0" durumuna geçerse ve eğer durum göstericisindeki kesme maskeleme biti "1" değilse (yani kesmelere izin verilmişse) mikroişlemci bir kesme çevrimine girer. Fakat önce yürütmekte olduğu komutu tamamlar ve sonra program sayıcısını, indis göstericisini, akümülatörleri ve durum göstericisini yığına atar. CCR (Condition code register)'nin (durum göstericisinin) I biti "1" durumuna getirilerek yeni bir kesme isteminin cevaplanması önlenir. Daha sonra da, kesme hizmet programının (kesme nedeniyle yürütülecek programın) adresini alabilmek için adres taşıtına FFF8 ve FFF9 çıkarılır. Bu bellek konumlarındaki iki bayt'lık bilgi kesme hizmet yordamının başlangıç adresidir, dolayısı ile PC'ye (program sayıcısına) yüklenir. FFF8 ve FFF9'a kesme vektörü (IRQ vector) adı verilir. Kesmelerin cevaplanabilmesi için, $\overline{\text{HALT}}$ ucunun "1" olması gerektiği unutulmamalıdır.

$\overline{\text{NMI}}$ (Non-Maskable Interrupt - Maskelenemeyen kesme): 6802 Mikroişlemcisi iki tür kesme kabul edilebilir: Bunlardan bir tanesi, CCR (Durum saklayıcısı)'nın I bitiyle maskelenebilir. Diğeri ise, bu bitten etkilenmeyen (yani maskelenemeyen) bir kesme olup daha çok güç kesilmesi veya mikroişlemcinin mutlaka hemen cevap vermesi gereken durumlar için kullanılır. $\overline{\text{NMI}}$ girişinin "0" düzeyine inmesi mikroişlemcinin derhal (yürütmekte olduğu komutu bile bitirmeden) PC, IX, akümülatörler ve CCR'e yığına itmesine ve adres taşıtına $\overline{\text{NMI}}$ vektörü olan FFFC ve FFFD'yi çıkarmasına neden olur. Bu bellek konumundaki veriler, maskelenemeyen kesme servis alt programının başlangıç adresi olarak PC'ye yüklenir.

6800 ailesinde, yazılım kesmesi olarak adlandırılan SWI bir kesme daha vardır ki, bu komut da mikroişlemci, yine durumunu yığına attıktan sonra gideceği servis yordamının adresini yazılım kesme vektörü olarak bilinen FFFA ve FFFB bellek konumlarından okur.

VMA (Valid Memory Address - Geçerli bellek adresi): Bu çıkış "1" durumunda olduğu zaman mikroişlemci; çevresindeki bütün birimlere adres taşıtında geçerli bir adres olduğunu bildirir. Bazı komutların yürütülmesinde adres taşıtı, rasgele değerler taşıyabilir. Herhangi bir dış birimin bu nedenden dolayı hatalı bir şekilde etkinleştirilmesi olasılığını ortadan kaldırmak için VMA, hattı bu gibi durumlarda "0" düzeyinde tutulur. VMA, HALT veya WAI komutunun yürütülmesi sırasında da "0" durumundadır.

$\overline{\text{HALT}}$ (Bekle): 6802'nin bu girişi, "1" durumunda ise mikroişlemci RUN (çalışma) modundadır ve komutlar yürütülmektedir. $\overline{\text{HALT}}$ hattı "0" a çekildiği zaman, mikroişlemci, yürütmekte olduğu komutu bitirir ve BEKLE moduna geçer. Bu durum da BA hattı "1", VMA hattı "0" düzeyinde, bütün 3 durumlu hatlar ise yüksek empedans durumunda olur. Böylece mikroişlemci, bütün sistemden yalıtılmış olur.

ENABLE: Bu çıkış, mikroişlemci ile dış birimler arasında senkron çalışmayı temin eder. Bu çıkış işaretlerinin frekansı, bir makina çevrim frekansıdır.

MR (Memory Ready - Bellek Hazır): Erişim süreleri uzun olan bellek elemanlarına, bilginin kaydedilmesi için bir çıkıştır.

RE (RAM Enable - RAM bellek kullanılabilir): 6802 mikroişlemcisinin $0000 -007F adreslerinde bulunan RAM bellek'in seçilmesinde kullanılır. Adres çözücüsü, 0000 - 007F adresini çözüp bu ucu etkin hale getirmelidir.

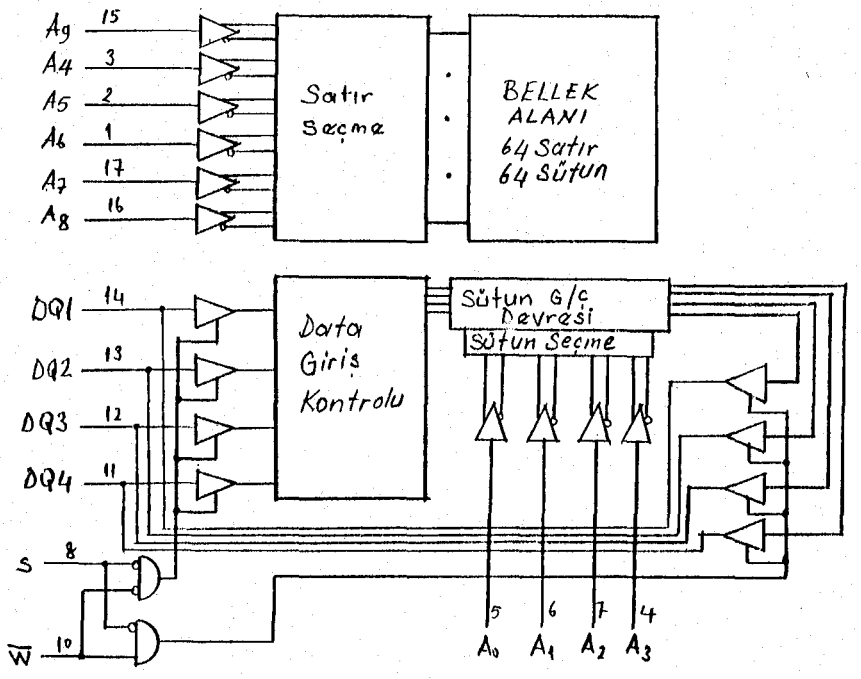
V_{CC} Stanby: Mikroişlemci içersindeki RAM belleğin, enerji kesilmesine karşın $0000 ile 001F hafıza bölgesindeki bilgilerin kaybolmaması için 5V'luk besleme ucudur.

4.2- 2114 RAM Bellek:

2114 RAM bellek 4096-bit rastgele erişimli statik bellektir. 18 ayaklı bir muhafaza içersinde 4 bitliktir. (1024 x 4). Bu tür bir RAM belleğin ayakları aşağıda açıklanmıştır.

-(A_0-A_9), Adres Taşıtı Uçları
-(D_0-D_3), Veri Taşıtı Uçları
-$\overline{W}$ Yazmayı etkinleştirme ucu
-$\overline{S}$, RAM entegrenin seçme ucu
-V_{ss} Toprak

2114 tipi RAM belleğin yapısı şekil 4.2.'deki gibidir.



Şekil 4.2. 2114 RAM Bellek Yapısı.

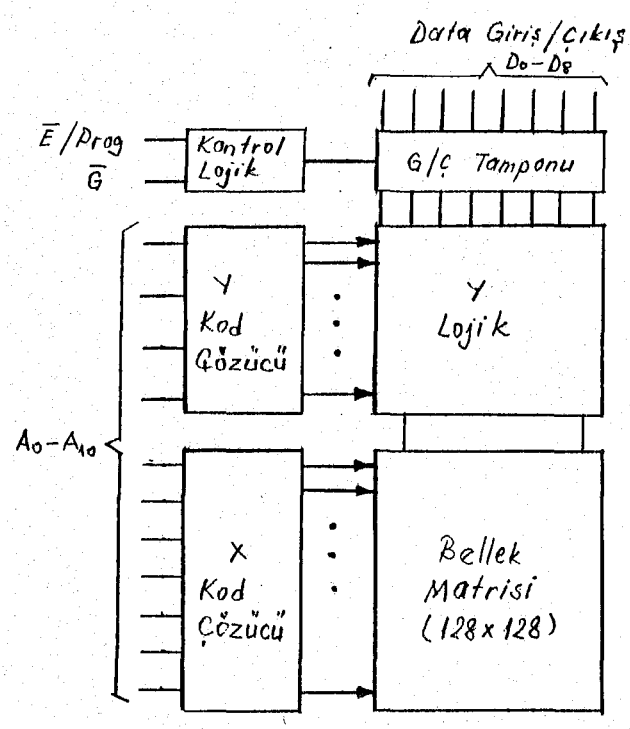
4.3 2716 EPROM Bellek

2716 Ultraviyole ışınlarla silinebilen ve elektriksel olarak programlanabilen 2048x8 Bitlik PROM bellektir. Tek besleme kaynağı ile beslenebilir (5 Volt). 2716 EPROM programlanabilmesi, PRG ayağının 25 V'luk sinyallerin uygulanması ile, bilgi taşıtındaki Veriyi ilgili adres gözüne yazarak yapılır. Bu nedenle bu tür bellekleri programlamak için ayrı bir düzenek gereklidir.

2716 EPROM bellek sistemde yazılımın saklatılması için düşünülmüştür ve yapısı şekil 4.3 deki gibidir.

2716 Bellek ayakları aşağıda açıklandığı gibidir:

-A_0 - A_{10} Adres taşıtı
-D_0 - D_7 Veri taşıtı,
-E/PRGR Seçme / Programlama uçları
-G Çıkışa izin verme ayağı



4.3. - 2716 EPROM Bellek Yapısı.

4.4. 6821 PIA (Perhiperal Interface adapter - Çevre arabirimi)

6821 PIA, mikroişlemci ile çevre birimleri arasındaki paralel (sözçükler halinde) veri alış verişi için geliştirilmiş bir birimdir. Şekil 4.4'de verilen öbek şemada görüldüğü gibi A ve B olarak tanımlanmış birbirinin benzeri iki bölümden oluşur. Her bölümde şu üç saklayıcı vardır,

Denetim/Durum Saklayıcısı (Control Register): PIA'nın kendi tarafından (A veya B) çalışmasını denetler.

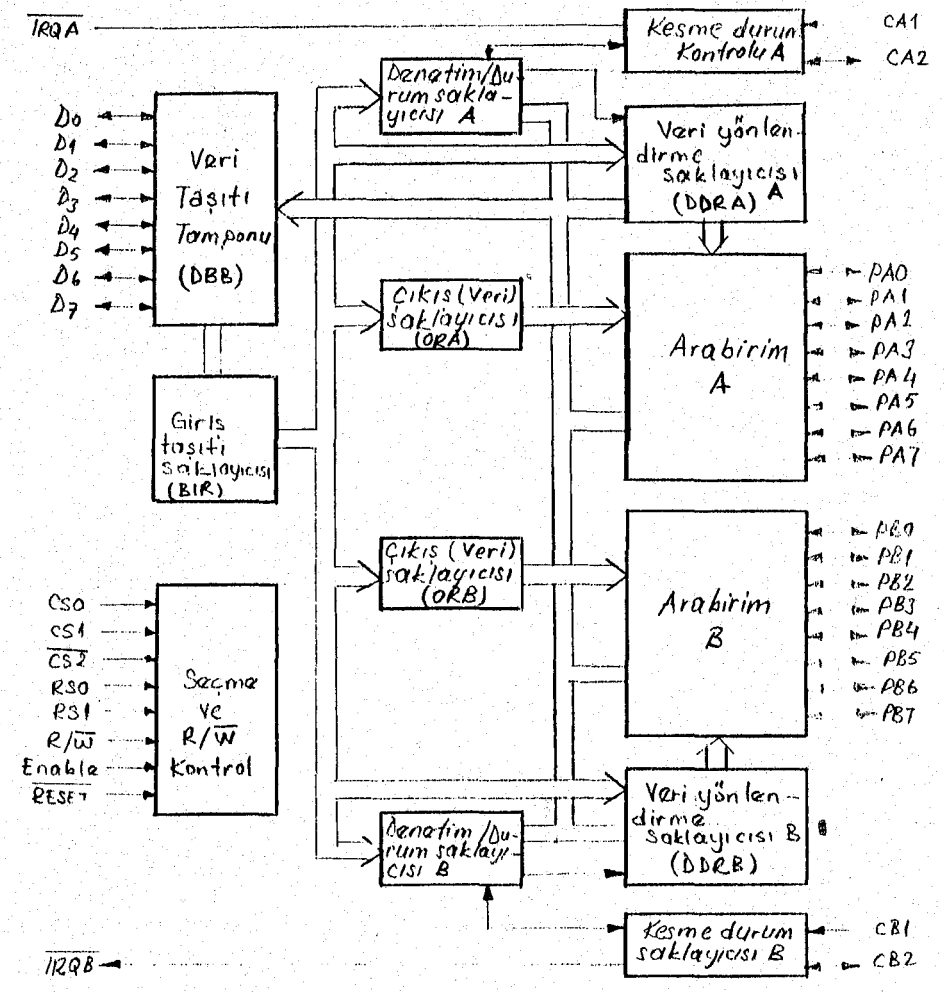
Veri Yönlendirme Saklayıcısı (Data Direction Register): 2 x 8 giriş/çıkış hattının hangilerinin çıkış olarak kullanılacağını belirler.

Çıkış (Veri) Saklayıcısı (Output Register): Dış birimden PIA'ya giren veya PIA'dan dış birimine giden veriler burada tutulur.

PIA saklayıcıları 6802 için birer bellek konumu gibidirler. 6802 bu saklayıcılar içine diğer bellek konumlarına yapacağı gibi yazar veya buralardan okur.Dolayısıylada sistem kurulurken bir ROM veya RAM belleğe adres verilmesi gibi PIA'ya belirli bir adres verilmesi gerekir.

PIA'nın ayak bağlantıları ve fonksiyonları aşağıda açıklanmıştır:

D_0 - D_7 Girişleri: Mikroişlemcinin veri taşıtına bağlanır.
RS0, RS1 ve CS0, CS1 ve $\overline{CS2}$ girişleri: Bu ayaklar uygun adres



Şekil 4.4. 6821 PIA'nın öbek şeması.

taşıtına bağlanır. RS0 ve RS1, A0 ve A1'e bağlanarak denetim saklayıcısının bir bitine bağlı olarak 6 saklayıcıdan hangisinin seçileceğini belirler. CS0, CS1 ve $\overline{CS2}$'nün bağlandığı adres hatları, PIA'nın adresini belirler.

E (Enable - Etkinleştirme) giriş: 6821 dinamik bir birim olduğundan E girişi, sürekli olarak saatlendirmelidir. Dolayısıyla 6802'nin E ucu ile birleştirmelidir.

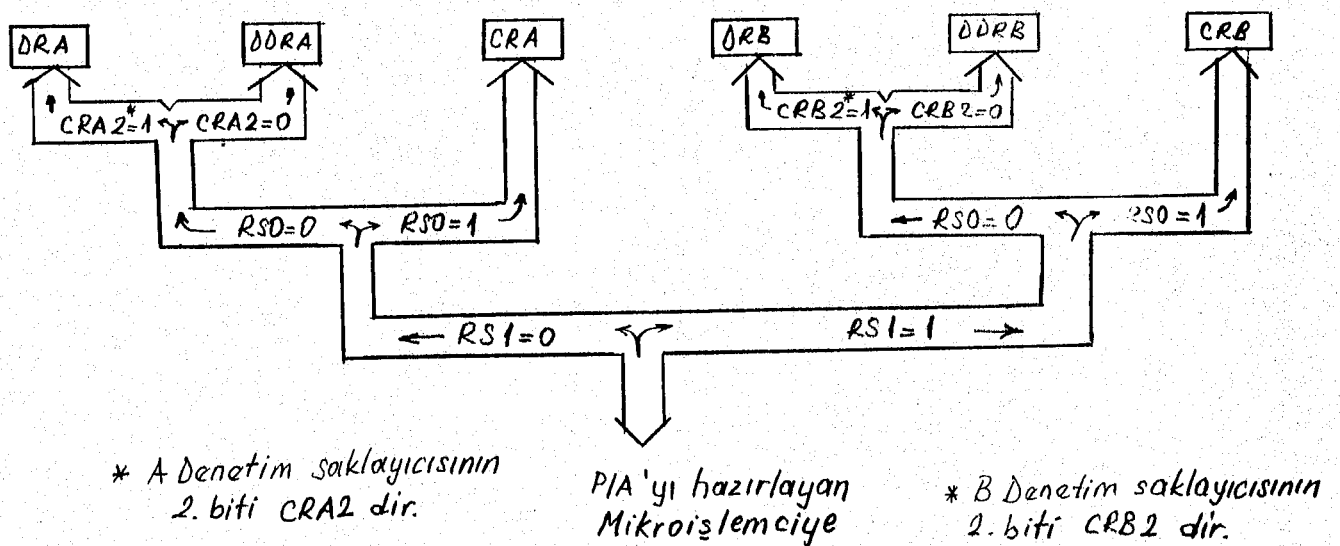
$R/\overline{W}$ (Read/Write - Oku/Yaz) girişi: Veri taşıtındaki verinin yönünü belirler. $R/\overline{W}$ "0" ise, giriş tamponları etkinleşir ve mikroişlemciden PIA'şa, E işareti sırasında, (ve eğer birim seçilmişse) veri aktarılır. $R/\overline{W}$ "1" ise, PIA mikroişlemciye veri aktarmaya hazır duruma geçer ve adreslenmiş olan saklayıcıdaki veri, E işareti ile birlikte veri taşıtına çıkarılır. PIA'nın bu ucu, mikroişlemcinin $R/\overline{W}$ ucu ile birleştirilmelidir.

RESET (Sıfırlama) girişi: Bu giriş "0" olduğu zaman, PIA saklayıcılarının bütün bitleri sıfırlanır. Genellikle, mikroişlemcinin $\overline{\text{RESET}}$ hattına bağlanır.

$\overline{\text{IRQA}}$ ve $\overline{\text{IRQB}}$ (Interrupt Request - Kesme İstemi A ve B): PIA'nın kesme hatlarıdır. Genellikle birleştirilerek 6802'nin $\overline{\text{IRQ}}$ girişine bağlanır.

PIA ile dış birimler arasındaki bağlantı hatları ise, sekizer bitlik iki grup (PAØ - PA7 ve PBØ-PB7) çift yönlü veri hatları ile her kısım için iki tane olmak üzere dört denetim hattıdır. (CA1 ve CA2 ile CB1 ve CB2). Bu denetim hatları, PIA ile dış birimler arasındaki iletişimin gerekli şekilde gerçekleşmesini sağlamakta kullanılır.

4.4.1- PIA'nın çalışmaya hazırlanması:



Şekil 4.5. PIA'nın çalışmaya başlatılması ve saklayıcılarının seçilmesi

PIA aracılığı ile veri alış verişine başlanmadan önce, denetim ve veri yönlendirme saklayıcılarının içerikleri, çalışma modunu ve veri yönünün belirleyecek bir duruma getirmek gerekir. Bu işlem başlangıçta yapılır ve anılan saklayıcıların içeriklerine genellikle bir daha dokunulmaz. Veri yönlendirme saklayıcıları (DDRA ve DDRB), 8 bit uzunluğundadır. Bu saklayıcıların

Bu saklayıcıların içeriği,portların G/Ç nı düzenler.Bir saklayıcı biti"1"ise ona karşı gelen PA (veya PB) hattı çıkış, "O" ise, giriş olur. Örneğin, başlangıçta DDRA'şa Ø yüklenirse, PA7-PA4 çıkış, PA3-PAO ise, giriş olarak tanımlanmış olur.

Çıkış saklayıcılarında (DRA ve DRB) içeri alınan veya dışarı verilecek olan veri bulunur. Çıkış saklayıcıları ve veri yönlendirme saklayıcıları aynı adrese sahiptir. Veri taşıtındaki bilginin hangi saklayıcıya alınacağı, denetim saklayıcısının bit 2'sinin durumuna bağlıdır.

Tasarımı yapılan mikrobilgisayarın 1. PIA'sının çalışmaya başlatılması işini, inceliyelim:

Program:

```
CLR  8ØØ1        8ØØØ de  RSØ = O
CLR  8ØØ3                 RS1 = O
LDA A #FØ        8ØØ1 de  RSØ = 1
STA A 8ØØ2                RS1 = O
LDA A #FF        8ØØ2 de  RSØ = O
STA A 8ØØØ                RS1 = 1
LDA A #3C        8ØØ3 de  RSØ = 1
STA A 8ØØ1                RS1 = 1
STA A 8ØØ3
```

Buna göre önce, CLR 8ØØ1 ve CLR 8003 ile CRA ve CRB'ye ulaşılıyor ve içerikleri sıfırlanıyor. Sonra, 8ØØ2 ye (DDRB) ye FØ yükleniyor. Bu şekilde B portunda PBØ...PB3 giriş, PB4...PB7 çıkış olacak şekilde seçiliyor. 8ØØ4'e (DDRA) ya FF yüklenerek Aportunun çıkış olması (PAO...PA7) sağlanıyor. 8ØØ1 ve 8ØØ3 adreslerine ØØC yüklenerek, A ve B çıkış saklayıcıları seçilmiş oluyor. Aynı zamanda, kesme istemi de önlenmiş oluyor.

Bu mikrobilgisayarda 1. PIA'ya 8ØØØ...8ØØØ, 2.PIA'ya AØØØ ...A003 adresleri tahsis edilmiştir. 1.PIA Displaylerin çoğullamasında ve keypad için kullanılmıştır. 2. PIA kontrol giriş/çıkış için kullanılmıştır.

4.5 Birimlerin ve bellek bölümünün adreslenmesi

Mikrobilgisayar donanım şemasından (Şek 4.6) görüldüğü gibi, adresleme işini, 74139 kod çözücü yapmaktadır. Bu tümleşik devre, iki ayrık kısımdan olmuştur. A ve B girişlerinin değerlerine göre, 4 çıkıştan birini, "O" yapacak şekilde çalışmaktadır. A ve B girişleri A13 ve A14 adres hatları ile birleş-

tirilmiştir. Tümleşik devrenin ayrık 1. kısmını A15, 2. kısmını $\overline{A15}$ etkinleştirmektedir. Buna göre,

Adres taşıtı	A13	A14	A15	
ø 0000	0	0	0	1. Taraf etkin 4 no'lu uç "0"
ø 1FFF	0	0	0	" " " " "
ø 2000	1	0	0	" " " 5 no'lu uç "0"
ø 3FFF	1	0	0	" " " " " "
ø 4000	0	1	0	" " " 6 no'lu uç "0"
ø 5FFF	0	1	0	" " " " "
ø 6000	1	1	0	" " " 7 no'lu uç "0"
ø 7FFF	1	1	0	" " " " "
ø 8000	0	0	1	2. Taraf etkin 12 no'lu uç "0"
ø 9FFF	0	0	1	" " " "
ø A000	1	0	1	2. Taraf etkin 11 no'lu uç "0"
ø BFFF	1	0	1	" " " "
ø C000	0	1	1	2. Taraf etkin 10 no'lu uç "0"
ø DFFF	0	1	1	" " " "
ø E000	1	1	1	2. Taraf etkin 9 no'lu uç "0"
ø FFFF	1	1	1	" " " "

olmaktadır.

Sonuç olarak;

0000 - 1FFF arasında 4 nolu uca bağlı CPU'nun içindeki RAM seçilir.

2000 - 3FFF arasında 5 nolu uca bağlı 2x2114 tipi RAM seçilir

4000 - 5FFF arasında 6 nolu uç (boş) seçilir.

6000 - 7FFF arasında 7 nolu uç (boş) seçilir.

8000 - 9FFF arasında 12 nolu uca bağlı 1. PIA seçilir.

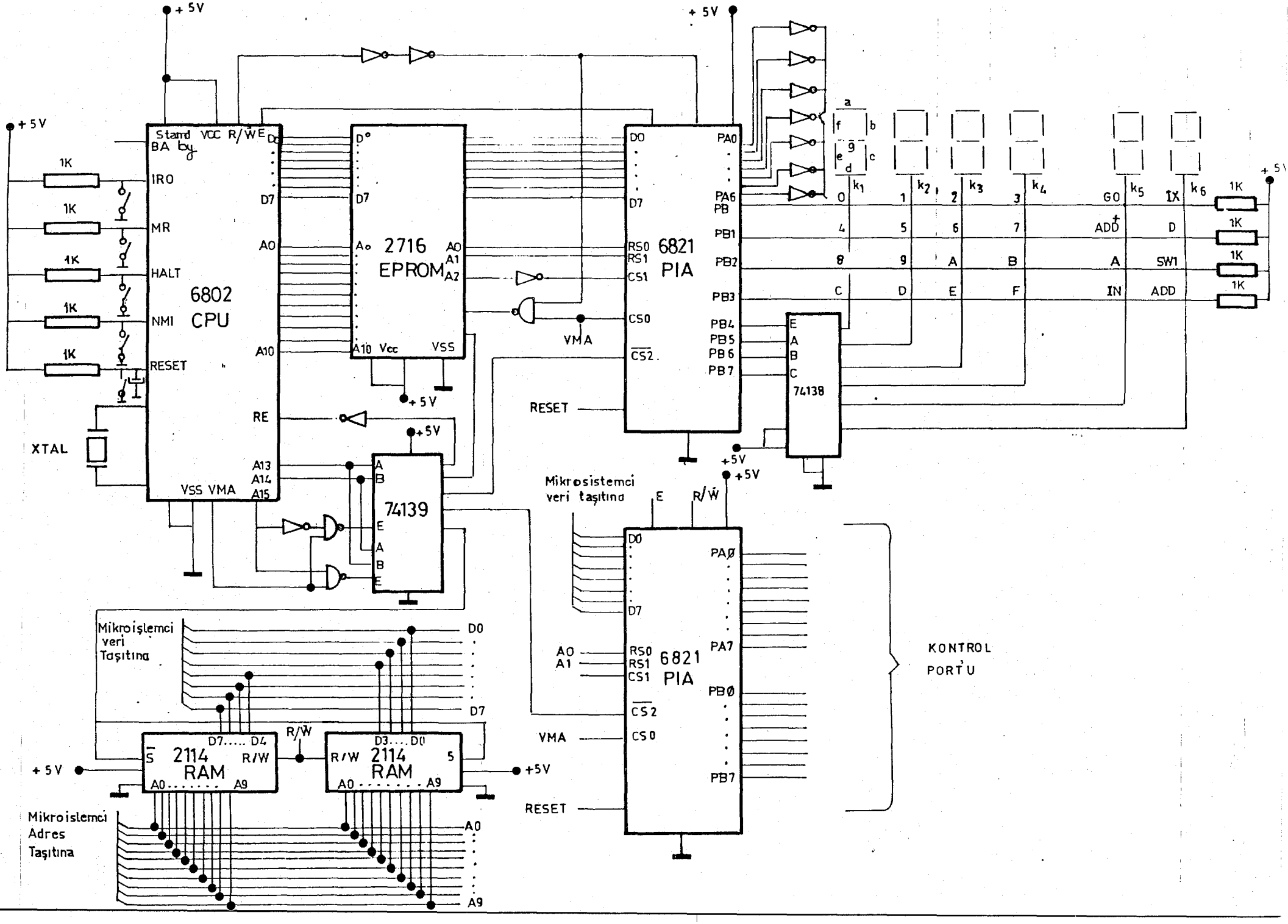
A000 - BFFF arasında 11 nolu uca bağlı 2. PIA seçilir.

C000 - DFFF arasında 10 nolu uç (boş) seçilir.

E000 - FFFF arasında 9 nolu uca bağlı 2716 EPROM seçilir.

6802 mikroişlemcide RESET vektörü FFFE - FFFF adres gözleri olduğundan en yüksek değerli adres gözlerine, genellikle sistem yazılımı (Software) konur.

Bu adresleme ile mikroişlemcinin RAM'ı ØØØØ ile 007F arasında, 2ØØØ ile 3FFF arasında da 2114 RAM bulunmaktadır.


+5V
Stand by
VCC
R/W
BA
IRQ
MR
HALT
NMI
RESET
6802 CPU
XTAL
VSS
VMA
D0
D7
A0
A10
RE
A13
A14
A15
2716 EPROM
A1
A2
Vcc
VSS
1K
74139
6821 PIA
RS0
RS1
CS1
CS0
CS2
PA0
PA6
PB
PB1
PB2
PB3
PB4
PB5
PB6
PB7
74138
a
b
c
d
e
f
g
k1
k2
k3
k4
k5
k6
0
1
2
3
4
5
6
7
8
9
A
B
C
D
E
F
GO
ADD
DA
IX
SW1
IN
Mikrosistemci veri taşıtına
E
R/W
PA0
PA7
PB0
PB7
KONTROL PORTU
Mikroişlemci veri Taşıtına
2114 RAM
D7..... D4
D3..... D0
Mikroişlemci Adres Taşıtına
A0
A9

5. MİKROİŞLEMCİLİ MİKROBİLGİSAYARIN YAZILIMI (SOFTWARE)

Bu tez çalışmasında 4. bölümde tasarlanan mikrobilgisayar genel amaçlıdır. Tezin konusu olan bir baskı devre hattı modelinin kontrolu, bu genel amaçlı mikrobilgisayarla uygulamadır.

Baskı devre kaplama hattı modelinin kontrolu için gerekli yazılım "kaypad" tuşları ile RAM belleğe kaydedilecek ve yazılım display'den görülebilecektir.

Genel amaçlı mikrobilgisayar "Keypad" ve "Display" için sistem yazılımı bu bölümün konusudur. Display'lerden soldan dördü bellek adreslerini hexadecimal olarak göstermektedir, diğer ikisi dört display'de görülen adresdeki veri ve dataları göstermektedir.

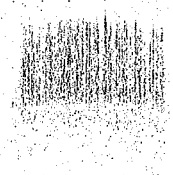
Sistem yazılımı oluştururken yığın için (SP), program sayısı icin (PC), indis saklayıcısı için (IX), A ve B akümülatörleri için 6802 mikroişlemcisinin içerisinde bulunan RAM belleğin $ 007F den $0059'a kadar olan kısmı (Tamamı $ØØØØ dan $ØØ7F'e kadar) sistem yazılımı için kullanılmıştır.

"Kaypad" tuş takımında, Ø dan F'e kadar onaltılık sayı sistemi elemanları, RUN tuşu,
ADD+ tuşu,
ADD- tuşu,
A akümülatörü tuşu,
B akümülatörü tuşu,
SWI tuşu,
IN tuşu,
IX tuşu,
RESET tuşu bulunmaktadır.

./..

Bu tuşlar ve işlevleri aşağıdaki gibidir;

Ø....F	: Program girerken makina kodu için gerekli onaltılık sayı sistemi elemanlarıdır.
RUN	: Program tamamlanıp, çalışmaya başlatmak için kullanılır.
ADD+	: Adresleri birer birer artırır. Program yazımında kolaylık sağlar.
ADD-	: Adresleri birer birer azaltır. Program yazımında ve yazılan programın kontolunda kolaylık sağlar.
A akümülatör	: A akümülatörü içerisindeki bilginin okunmasını sağlar.
B akümülatör	: B akümülatörü içerisindeki bilginin okunmasını sağlar.
SWI	: Daha önce yazılım ilkelerinde bahsettiğimiz yazılım kesmesi tuşudur. O anda display' lerin gösterdiği adres gözüne yazılım kesmesi komutunun makina kodu olan 3F'i yazar.
IN	: Bu tuş ile display'lerin gösterdiği adres gözüne (RAM'a ait) bilginin kaydedilmesi işini yapar.
IX	: İndis saklayıcısının içerdiği bilgiyi görmek için kullanılır.
RESET	: Mikrobilgisayarı Reset'lemede kullanılır.



./..

Sistem yazılımında dört altprogram düşünülmüştür. SUB0, SUB1 SUB2, SUB3. SUB0 altprogramı "keypad" ile program yazmak içindir. Diğer altprogramlar yedek olarak düşünülmüş bu şekilde yazılım geliştirmeye uygun tutulmuştur. Altprogram seçimi, tuşların sırasına uygun biçimdedir. Makinayı RESET'ledikten sonra 1. sıradaki tuşlardan herhangi birine basılırsa SUB0,

2. sıradaki tuşlardan herhangi birine basılırsa SUB1 seçilmiş olur.

Donanım şemasından (Şekil 4.6) görüleceği gibi 1. PIA'nın B portunda PB4, PB5, PB6, PB7 display seçmek için dört bit mevcuttur. Oysa seçilmesi gereken altı display vardır. Bu nedenle 4'den 6'ya kod çözümü 74138 kullanılmış ve yazılım buna göre düzenlenmiştir.

Yazılım başlangıç adresi $FFFE (E0), FFFF (16) olarak (FFFE ve FFFF Reset Vektörüdür.) yazılmıştır. Program $E016 adresinden başlamaktadır.

Bu aşamada, kesme işlemi gerekmediğinden IRQ, NMI, HALT gibi uçlar "1" değerindedir. Ancak kullanıma olanak vermek için, anahtarla etkin duruma getirilebilir.

Tasarımı yapılan mikrobilgisayarın sistem yazılımı EK 1 dedir.

./..

PROGRAM : 6802 MİKROİŞLEMCİSİ İÇİN SİSTEM YAZILIMI SAYFA 1

Nr		Sembolik Dil	Operator	Adres	Mak. Kod		AÇIKLAMA
		NOP		FE16	01		
		CLR	8001	17	7F	8001	
		CLR	8003	1A	7F	8003	
		LDA A	# F0	1D	86	F0	
		STA A	8002	1F	B7	8002	
		LDA A	# FF	22	86	FF	
		STA A	8000	24	B7	8000	
		LDA A	# 3C	27	86	3C	
		STA A	8001	29	B7	8001	
		STA A	8003	2C	B7	8003	
		JSR	FD80	2F	7E	FD80	
		NOP		32	01		
		BRA	0F	33	20	0F	
		BRA	28	35	20	28	
		NOP		FE44	01		
		STS	$72	45	9F	72	
		PUL A		47	32		
		LDS	$71	48	97	71	
		STA A	800C	4A	B7	800C	
		PUL A		4D	32		
		LDS	$70	4E	97	70	
		PUL A		50	32		
		LDS	$6F	51	97	51	
		STX	$74	53	DF	74	
		LDX	$6A	55	DE	6A	
		DEC X		57	09		
		LDA A	$6E	58	96	6E	
		STA A	X,00	5A	A7	00	
		INC X		5C	08		
		STX	$6C	5D	DF	6C	
		NOP		5F	01		
		LDS	# 007F	60	8E	007F	
		LDA A	# FE	63	86	FE	
		STA A	$77	65	97	77	
		LDX	$6C	67	DE	6C	

PROGRAM :

SAYFA 2

Nr		Sembolik Dil	Operator	Adres	Mak. Kod		AÇIKLAMA
		LDA A	X,00	FE69	A6	00	
		AND A	# OF	68	84	OF	
		ADD A	# B6	60	8B	B6	
		STA A	$ 78	6F	97	78	
		LDX $	77	71	DE	77	
		LDA A	X,00	73	A6	00	
		PSH A		75	36		
		LDX $	6C	76	DE	6C	
		LDA A	X,00	78	A6	00	
		AND A	# FO	7A	84	FO	
		LSR A		7C	44		
		LSR A		7D	44		
		LSR A		7E	44		
		LSR A		7F	44		
		ADD A	# B6	80	8B	B6	
		LDS	$ 78	82	97	78	
		LDX	$ 77	84	DE	77	
		LDA A	X,00	86	A6	00	
		PSH A		88	36		
		LDA B	# 02	89	C6	02	
		LDA A	$ 6D	8B	96	6D	
		LDS	$ 79	8D	97	79	
		AND A	# OF	8F	84	OF	
		ADD A	# B6	91	8B	B6	
		LDS	$ 78	93	97	78	
		LDX	$ 77	95	DE	77	
		LDA A	X,00	97	A6	00	
		PSH A		99	36		
		LDA A	$ 79	9A	96	79	
		AND A	# FO	9C	84	FO	
		LSR A		9E	44		
		LSR A		9F	44		
		LSR A		AO	44		
		LSR A		A1	44		
		ADD A	# B6	A2	8B	B6	
		LDS	$ 78	P4	97	78	
		LDX	$ 77	A6	DE	77	
		LDA A	X,00	A8	A6	00	
		PSH A		AA	36		
		DEC B		AB	5A		
		BNE	04	AC	26	04	
		LDX	$ 6C	AE	DE	6C	
		BRA 14		B0	20	14	
		LDA A	$ 6C	B2	96	6C	
		BRA	D7	B4	20	D7	
		"0"		B6	CO		
		"1"		B7	F9		
		"2"		B8	A4		

PROGRAM :

SAYFA 3

Nr		Sembolik Dil	Operator	Adres	Mak. Kod		AÇIKLAMA
		"3"		FEB9	B0		
		4		BA	99		
		5		BB	92		
		6		BC	82		
		7		BD	F8		
		8		BE	80		
		9		BF	90		
		A		CD	88		
		B		C1	83		
		C		C2	C6		
		D		C3	A1		
		E		C4	86		
		F		C5	8E		
		NOP		C6	01		
		BRA	07 (3)	C7	20	07	
				C9	PF		
				CA	FF		
				CB	FF		
				CC	FF		
				CD	FF		
				CE	FF		
				CF	FF		
	3	LDS	# 006B	DO	8E	006B	
		BSR	02	D3	8D	02	
		BRA	F9 LOOP	D5	20	F9	
		NOP		D7	01		
		LDS	# 0079	D8	8E	0079	
		CLR B		DB	5F		
		ADD B	# 20	DC	CB	20	
		PULA		DE	32		
		STA A	8000	DF	B7	8000	
		STA B	8002	E2	F7	8002	
		LDA A	# FF	E5	86	FF	
	LOOP 1	DEC A		E7	4A		
		BNE	FA LOOP1	E8	26	FD	
		DEC A		EA	4A		
		STA A	8000	EB	B7	8000	
	LOOP 2	DEC A		EE	4A		
		BNE	FD LOOP2	EF	26	FD	
	LOOP 3	DEC A		F1	4A		
		BNE	FD LOOP3	F2	26	FD	
		LDA A	8002	F4	B6	8002	
		COM A		F7	43		
		AND A	# OF	F8	84	OF	
		STA A	$ 76	FA	97	76	
		TBA		FC	17		
		LSR A		FD	44		
		LSR A		FE	44		

PROGRAM :

SAYFA 4

Nr		Sembolik Dil	Operator	Adres	Mak. Kod		AÇIKLAMA
		LSR A		FEFF	44		
		LSR A		FF00	44		
		LSR A		01	44		
		DEC A		02	4A		
		ADD A	# 5F	03	8B	5F	
		STA A	$ 78	05	97	78	
		CLR	0077	07	7F	0077	
		LDX	$ 77	0A	DE	77	
		LDA A	$ 76	0C	96	76	
		BEQ	06 LOOP	0E	27	06	
		LDA A	X,00	10	A6	00	
		BEQ	0A	12	27	0A	
		BRA	0E	14	20	0E	
	LOOP	LDA A	X.00	16	A6	00	
		BEQ	0A (1)	18	27	0A	
		CLR	X,00	1A	6F	00	
		BRA	06 (1)	1C	20	06	
		LDA A	$ 76	1E	96	76	
		STA A	X,00	20	A7	00	
		BRA	0A (2)	22	20	0A	
	1	CMP B	# C0	24	C1	C0	
		BEQ	02 (3)	26	27	02	
		BRA	B2	28	20	B2	
	3	LDS	# 0069	2A	8E	0069	
	2	RTS		2D	39		
		CMP A	# 01	2E	81	01	
		BEQ	0C	20	27	0C	
		CMP A	# 02	32	81	02	
		BEQ	08	34	27	08	
		CMP A	# 04	36	81	04	
		BEQ	04	38	27	04	
		CMP A	# 08	3A	81	08	
		BNE	E6	3C	26	E6	
		LDX	$5D	3E	DE	5D	
		JMP	X,00	40	6E	00	
		NOP		42	01		
		LDX	$ 59	43	DE	59	
		CMP B	# A0	45	C1	A0	
		BNE	25	47	26	25	
		LDAA	$ 76	49	96	76	
		CMP A	# 01	43	81	01	
		BEQ	0A	4D	27	0A	
		CMP A	# 02	4F	81	02	
		BEQ	08	51	27	08	
		CMP A	# 04	53	81	04	
		BEQ	09	55	27	09	
		BRA	0D	57	20	0D	
		JMP	X,00	59	6E	00	

PROGRAM :

SAYFA 5

Nr		Sembolik Dil	Operator	Adres	Mak. Kod		AÇIKLAMA
		LDX	$ 6C	FF5B	DE	6C	
		INX		5D	08		
		BRA	3B	5E	20	3B	
		NOP		60	01		
		LDS	# 006F	61	CE	006F	
		BRA	35	64	20	35	
		NOP		66	01		
		LDA A	$ 6D	67	96	6D	
		STA A	X.00	69	A7	00	
		INX		6B	08		
		BRA	2D	6C	20	2D	
		NOP		6E	01		
		CMP A	# C0	6F	C1	C0	
		BNE	2F	71	26	2F	
		LDA A	$ 76	73	96	76	
		CMP A	# 01	75	81	01	
		BEQ	0A	77	27	0A	
		CMP A	# 02	79	81	02	
		BEQ	0B	7B	27	0B	
		CMP A	# 04	7D	81	04	
		BEQ	0D	7F	27	0D	
		BRA	16	81	20	16	
		NOP		83	01		
		LDX	$ 74	84	DE	74	
		BRA	13	86	20	13	
		NOP		88	01		
		LDX	# 0070	89	CE	0070	
		BRA	0D	8C	20	0D	
		NOP		8E	01		
		LDA A	X.00	8F	A6	00	
		STA A	$ 6E	91	97	6E	
		LDA A	# 3F	93	86	3F	
		STA A	X.00	95	A7	00	
		BRA	02	97	20	02	
		NOP		99	01		
		DEC X		9A	09		
		STX	$ 59	9B	DF	59	
		STX	$ 6C	9D	DF	6C	
		JMP	FE5F	9F	7E	FE5F	
		NOP		A2	01		
		TBA		A3	17		
		LSR A		A4	44		
		LSR A		A5	44		
		LSR A		A6	44		
		LSR A		A7	44		
		LSR A		A8	44		
		DEC B		A9	4A		

PROGRAM :

SAYFA 6

Nr		Sembolik Dil	Operator	Adres	Mak. Kod		AÇIKLAMA
		STA A	$ 77	FF AA	97	77	
		LDA A	$ 76	AC	96	76	
		CMP A	# 01	AE	81	01	
		BNE	01	B0	26	04	
		LDA A	# 00	B2	86	00	
		BRA	12	B4	20	12	
		CMP A	# 02	B6	81	02	
		BNE	04	B8	26	04	
		LDD A	# 04	BA	86	04	
		BRA	0A	BC	20	0A	
		CMP A	# 04	BE	81	04	
		BNE	04	CD	26	04	
		LDA A	# 08	C2	86	08	
		BRA	02	C4	20	02	
		LDA A	# 0C	C6	86	0C	
		ADD A	$ 77	C8	98	77	
		STA A	$76	CA	97	76	
		LDA A	$ 6C	CC	96	6C	
		ASL A		CE	48		
		ASL A		CF	48		
		ASL A		DO	48		
		ASL A		D1	48		
		STA A	$ 6C	D2	97	6C	
		LDA A	$ 6D	D4	96	6D	
		LSR A		D6	44		
		LSR A		D7	44		
		LSR A		D8	44		
		LSR A		D9	44		
		ORA	$ 6C	DA	9A	6C	
		STA A	$ 6C	DC	97	6C	
		LDA A	$ 6D	DE	96	6D	
		ASL A		E0	48		
		ASL A		E1	48		
		ASL A		E2	48		
		ASL A		E3	48		
		ORA	$ 76	E4	9A	76	
		STA A	$ 6D	E6	97	6D	
		JMP	FF24	FF E8	7E	FF24	
		NOP		EB	01		
		LDX	$ 5B	EC	DE	5B	
		JMP	X,00	EE	6E	00	
				FO	FF		
				F1	FF		
				F2	FF		
				F3	FF		
				F4	FF		
				F5	FF		
				F6	FF		

PROGRAM :

SAYFA 7

Nr		Sembolik Dil.	Operator	Adres	Mak. Kod		AÇIKLAMA
		LDX.	# FD8A	FD80	CE	FD8A	
		STX	$5D	83	DF	5D	
		LDX	# FDF7	85	CE	FDF7	
		BRA	53 (1)	88	20	53	
		LDX	# FD94	8A	CE	FD94	
		STX	$ 5D	8D	DF	5D	
		LDX	# FDFD	8F	CE	FDFD	
		BRA	49 (2)	92	20	49	
		CMP A	# 08	94	81	08	
		BEQ	38 (3)	96	27	38	
		CMP A	# 04	98	81	04	
		BEQ	27 (4)	9A	27	27	
		CMP A	# 02	9C	81	02	
		BEQ	16 (5)	9E	27	16	
		LDX	# FDAA	A0	CE	FDAA	
		STX	$ 5D	A3	DF	5D	
		LDX	FE03	A5	CE	FE03	
		BRA	33 (6)	A8	20	33	
		LDX	# FFFF	AA	CE	FFFF	
		STX	$ 6C	FDAD	DF	6C	
		LDX	FF42	AF	CE	FF42	
		STX	$5D	B2	DF	5D	
		BRA	7F	B4	20	7F	
	5	LDX	# FDC0	B6	CE	FDC0	
		STX	$ 5D	B9	DF	5D	
		LDX	# FE09	FDBB	CE	FE09	
		BRA	1D	BE	20	1D	
		JMP	FFFF	C0	7E	FFFF	
	4	LDX	FDCD	FDC3	CE	FDCD	
	2	STX	$5D	C6	DF	5D	
		LDX	FE0F	C8	CE	FE0F	
		BRA	10	CB	20	10	
	1	JMP	FFFF	1 CD	7E	FFFF	
	3	LDX	# FDDA	EDD0	CE	FDDA	
		STX	$ 5D	D3	DF	5D	
		LDX	# FE15	D5	CE	FE15	
		BRA	03	D8	20	03	
		JMP	FFFF	DA	7E	FFFF	
	6	NOP		FDD	01		
		LDA B	# 06	DE	C6	06	
		LDS	# 007F	FDE0	8E	007F	
		LDA A	X, 00	E3	A6	00	
		PSH A		FDE5	36		
		DECX		E6	09		
		DEC B		E7	5A		
		BNE	F9	E8	26	F9	
		LDS	#006B	FDEA	8E	006B	

PROGRAM :

SAYFA 8

Nr		Sembolik Dil	Operator	Adres	Mak. Kod		AÇIKLAMA
		JSR	FED7	FDED	BD	FED7	
		BRA	F8	F0	20	F8	
		"G"					
		"t"		F2	87	05F2	
		"E"		F3	86		
		"F"		F4	8E		
		"_"		F5	BF		
		"0"		F6	CO		
		"1"		F7	F9		
				F8	92	E3	
				FA	83		
				FB	B7	FFFF	
				FE	92	E3	
				FE00	83		
				FE01	B7	FFCO	
				04	92	E3	
				06	83		
				07	B7	FFF9	
				0A	92	E3	
				0C	83		
				0D	B7	FFA4	
				10	92	E3	
				11	83		
				13	37	FFBO	

FFF8 FF
FFF9 FF } IRQ

FFFA FE
FFFB 32 } SWI

FFFC FE
FFFD EB } NMI

FFFE FE
FFFF 16 } RESET

6. BÖLÜM- KONTROL EDILECEK KIMYASAL KAPLAMA HATTI TASARIMLANMASI

6.1. Kimyasal kaplama hattı tasarımlanması:

Tez çalışmasının bu bölümünde 4. bölümde donanımı 5. bölümünde yazılımı anlatılan genel amaçlı mikrobilgisayarda kontrol edilecek kimyasal kaplama hattı anlatılacaktır. Hat çeşitli kimyasal işlemlerden geçecek (örneğin bakır kaplama) baskı devre plaketlerinin kimyasal işlem sırasına uygun olarak kontrol edilmesini amaçlamaktadır.

Baskı devre kaplama işlemi için bu bölümde modellenenden daha fazla banyo sayısı gerekecektir. Ancak 7. bölümde anlatılacak kontrol yazılımının temel algoritması değişmeyecektir. Bu tez çalışmasında benzetmenin daha pratik ve uygulanabilir olması açısından dört banyoyu içeren bir kimyasal kaplama hattı düşünülmüştür.

Hattın temel elemanları plaket tutucularının üzerine asılıp, işlem sırasına uygun, banyolara girip çıkmasını sağlayan taşıyıcı, banyolar, banyo konum sezicileri, kontrol mikrobilgisayarı ve içinde kimyasalların bulunduğu banyoların oturtulduğu mekanik konstrüksiyonlardır. Ayrıca her bir banyonun (eğer gerekiyorsa) ısı kontrolu, seviye kontrolu gibi kontrol.ar, kontrol bilgisayarından bağımsız elektromekanik yöntemlerle yapılmaktadır.

Bu türden kaplama hatları günümüzde elektronik sanayinde, baskılı devre kaplama ve metal kaplama işlerinde oldukça yaygın kullanılmaktadır. Genellikle manuel yöntemlerle kontrol edilen bir taşıyıcı gerekli işlem sırasına uygun kimyasal banyolara taşınmaktadır.

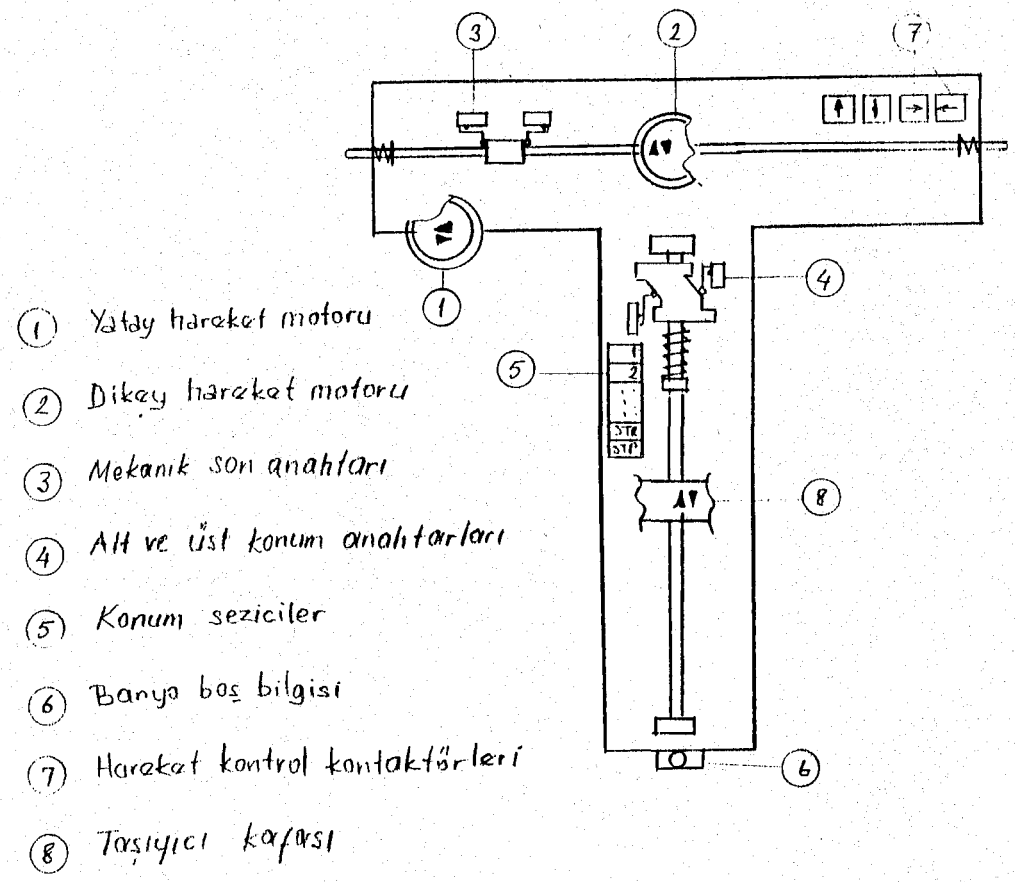
Diğer alt bölümlerde taşıyıcı elemanları anlatılmıştır.

6.2 Taşıyıcı

Taşıyıcı, kontrol mikrobilgisayarı tarafından kontrol edilen, sistemin hareketli olan parçasıdır. Taşıyıcı tutuculara asılmış bulunan plaketleri, mikrobilgisayara verilmiş işlem sırasına uygun kimyasal bulunan banyolara taşır.

Taşıyıcı üzerinde biri yatay hareketi, diğeri dikey hareketi sağlayan iki motor bulunmaktadır. Yatay hareket motoru, hızının ayarlanabilir olması gerektiğinden DC motordur. Dikey hareketi asenkron motor sağlamaktadır. Taşıyıcının yatay hareketle geldiği banyo ekseninde en iyi konumda durmasını sağlamak D.C. motorun frenleme kolaylığı bakımından, ayrıca tercih edilebilir. Taşıyıcının üzerinde motorları kontrol eden kontaktörlerde bulun-

maktadır. Motorların kontrolu, mikrobilgisayarın interface tarafından bobinlerine kontrol edilen bu kontaktörlerle sağlanmıştır. Şekil 6.1.'de anlatılan taşıyıcı şekli görülmektedir. Şekilde 5 nolu notasyonla gösterilen konum sezici dedektörler, taşıyıcı üzerindedir. Banyo konum bilgisini kontroldan bilgisayara iletir. 4 nolu elemanlar, taşıyıcı dikey hareket eden kısmın alt ve üst konum bilgilerini kontrola iletirler. (1) notasyonu ile gösterilen yatay hareket motoru (2) ile gösterilen dikey hareket motorudur. (3) no'lu mekanik limit anahtarıdır. Taşıyıcının bir engele çarpması halinde kontrola "alarm" durumunu bildirir. 6 no'lu metal dedektörü banyo boş bilgisini kontrola iletmekte kullanılır. 8 no'lu notasyonla belirtilmiş eleman plaketlerin üzerine dizildiği ve dikey hareket edebilen taşıyıcı kafasıdır.



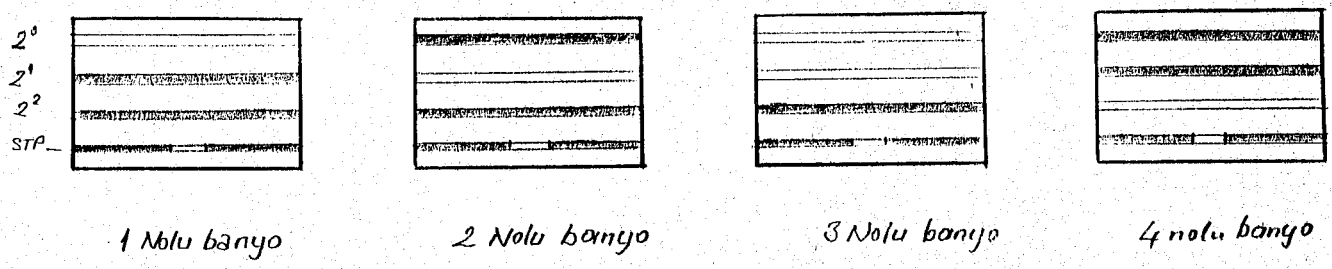
Şekil 1.6.1 Taşıyıcı ve elemanları

6.3. Banyo konum bilgilerinin sezilmesi

Taşıyıcının hangi banyoda olduğu ve tutuculara göre en iyi pozisyonun saptanması bu birim tarafından yapılır. Bu bilgilerin sezilmesi optik okuyucularla yapılır. Optik okuyucular, bir ışık kaynağının (-ki genellikle İnfrared LED diod ışık kaynağı olarak kullanılır) parlak bir yüzeye gönderilip, yansıyan ışığın algı-

lanması (fototransistor vasıtası ile) prensibi ile çalışır. Bu seziciler taşıyıcı üzerindedir. Işığın yansıtıldığı yüzeyler banyolar hizasında ve hattın üzerindedirler.

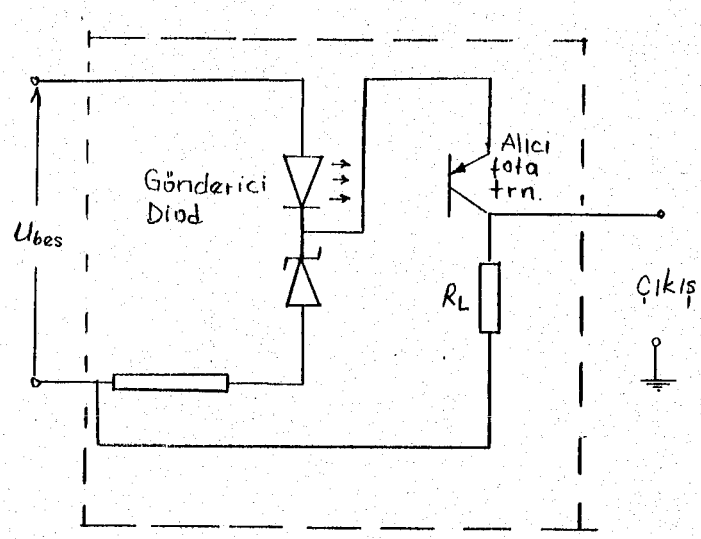
Konum bilgisi (banyo numaraları) ondalık kodlanmış ikili sayı sistemi ile düzenlenmiştir. Bu düzenleme ile dört adet banyo üç sezici kullanarak ifade edilebilir. Birinci sezici 2^0(1), ikincisi 2^1(2), üçüncüsü 2^2(4) değerlik taşır. (BCD sistem). Şekil 6.2'de bir, iki, üç ve dört numaralı banyoların konum sezme düzenleri gösterilmiştir.



Şekil 6.2 Banyo konum sezicileri için kullanılan yansıtıcı yüzeyleri.

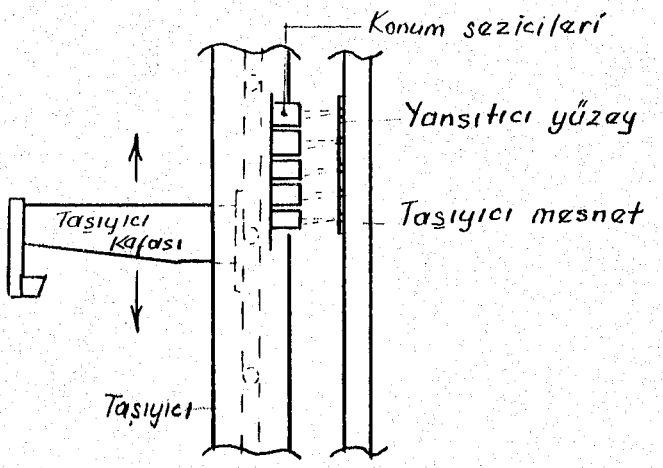
Şekil 6.2'de görülen STOP işareti bilgisi, taşıyıcının banyonun orta kısmında (en iyi pozisyonda) durmasını sağlar.

Şekil 6.3'de optik sezicilerin elektriksel yapısı görülmektedir.



Şekil 6.3. Sezicilerin elektriksel yapısı.

Şekil 6.4'te sezicileri sistem üzerinde yansıtıcı yüzey düzenleriyle birlikte gösterilmiştir.

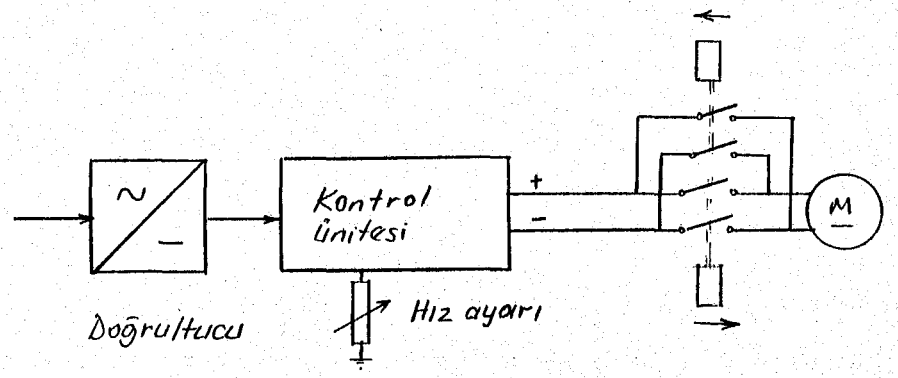


Şekil 6.4.- Seziciler ve yansıtıcılar

6.4. Taşıyıcı hareketinin oluşturulması.

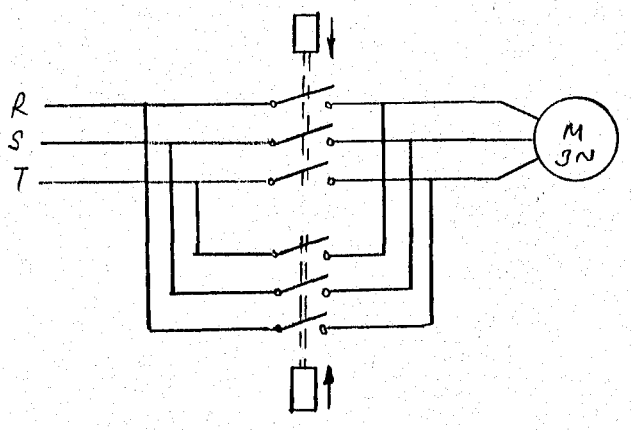
Taşıyıcı hareketinin oluşturulmasında birinci aşama 7. bölümde (alt bölüm 7.2'de) belirtilecek olan taşıyıcı hareket diagramının oluşturulmasıdır. Bu aşama kaplamanın kimyasal işlem sırası ile ilgili olduğundan kimya mühendisliğimi ilgilendirmektedir. Bu işlem sırası akış çizeneği biçiminde düzenlenmeli (Alt bölüm 7.3'de olduğu gibi) ve kontrol yazılımı oluşturularak kontrol mikrobilgisayarına girilmelidir. Kontrol yazılımı mikrobilgisayarda işlenip PIA'lar (Perhiperal Interface Adapter) tarafından bu bölümde anlatılan elemanlara iletilir. PIA'lar donanım özelliklerinden dolayı çıkış olarak 1,5 TTL yükü sürebilmektedir. Bu değer role ve kontaktör sürmek için oldukça küçüktür. Bu nedenle 500 mA'lık sürüçüler PIA ile kontaktörler arasında tampon olarak kullanılır.

Yatay hareketin kontrolu Şekil 6.5'de blok şeması çizildiği biçimdedir. Şekil 6.5'de D.C. Motorun frenlemesi gösterilmemiştir.



Şekil 6.5 - Yatay hareketin oluşturulması.

Dikey hareketin oluşturulmasında Şekil 6.6'daki bir donanımdan yararlanılır.



Şekil 6.6. Dikey hareketin oluşturulması.

Yatay ve dikey hareketin oluşturulmasında şekillerden görüldüğü gibi toplam dört kondaktör gerekmektedir. Bu kontaktörlerin bobinleri kontrol mikrobilgisayarı PIA'yı tarafından denetlenmektedir.

7. KONTROL BİLGİSAYARI DONANIM PROTOTİP VE YAZILIMI

Genel amaçlı olarak, tasarımlanmış 6802 mikroişlemcili mikrobilgisayarın kontrol edeceği 6. bölümde anlatılan bir kaplama hattını modelleyen prototip ve kontrol yazılımı bu bölümde incelenecektir.

7.1. Kontrol mikrobilgisayarı donanım, prototip

4. bölümde donanımı anlatılan mikrobilgisayarda 1.PIA display ve keypad için kullanıldığı görülmüştü. Yine donanım şemasından kontrol için 2. PIA kullanılacağı belirtilmişti. 2. PIA $A000 ile $A003 adreslerinde yer alacaktır. Kontrol yazılımı da $ 2000 adresinden başlayan RAM belleğe yazılacaktır. Kontrol yazılımı için gerekli alt programlar, 6802 mikroişlemci içinde yer alan 128 Byt'lik RAM belleğin ilk baytlarında yer alacak şekilde düzenlenmiştir. Kontrol yazılımı EK 2 de gösterilmiştir.

6. bölümde anlatılan bir kaplama hattının kontrolu için, giriş olarak aşağıdaki bilgilere ihtiyaç olacaktır.

2^0 2^1 2^2	Banyo konum bilgileri
Banyo bilgisi	- Banyoda taşıyıcı olup olmadığını kontrol edecek
Peryot	- Peryot devam bilgisi
Üst konum	- Taşıyıcının üst konuma geldiğini bildirecek
Alt konum	- Taşıyıcının alt konuma geldiğini bildirecek.

Bu giriş bilgilerini kontrol yazılımı ile işleyecek olan kontrol mikrobilgisayarı aşağıdaki çıkış bilgilerini verecek:

./..

- Taşıyıcı sağa komutu
- Taşıyıcı sola komutu
- Taşıyıcı yukarı komutu
- Taşıyıcı aşağı komutu

Alarm - Alarm komutu

Stop - Dur komutu

Bu giriş ve çıkış bilgileri 2. PIA'nın A ve B portları düzenlenmesi Tablo 7.1 de olduğu gibidir.

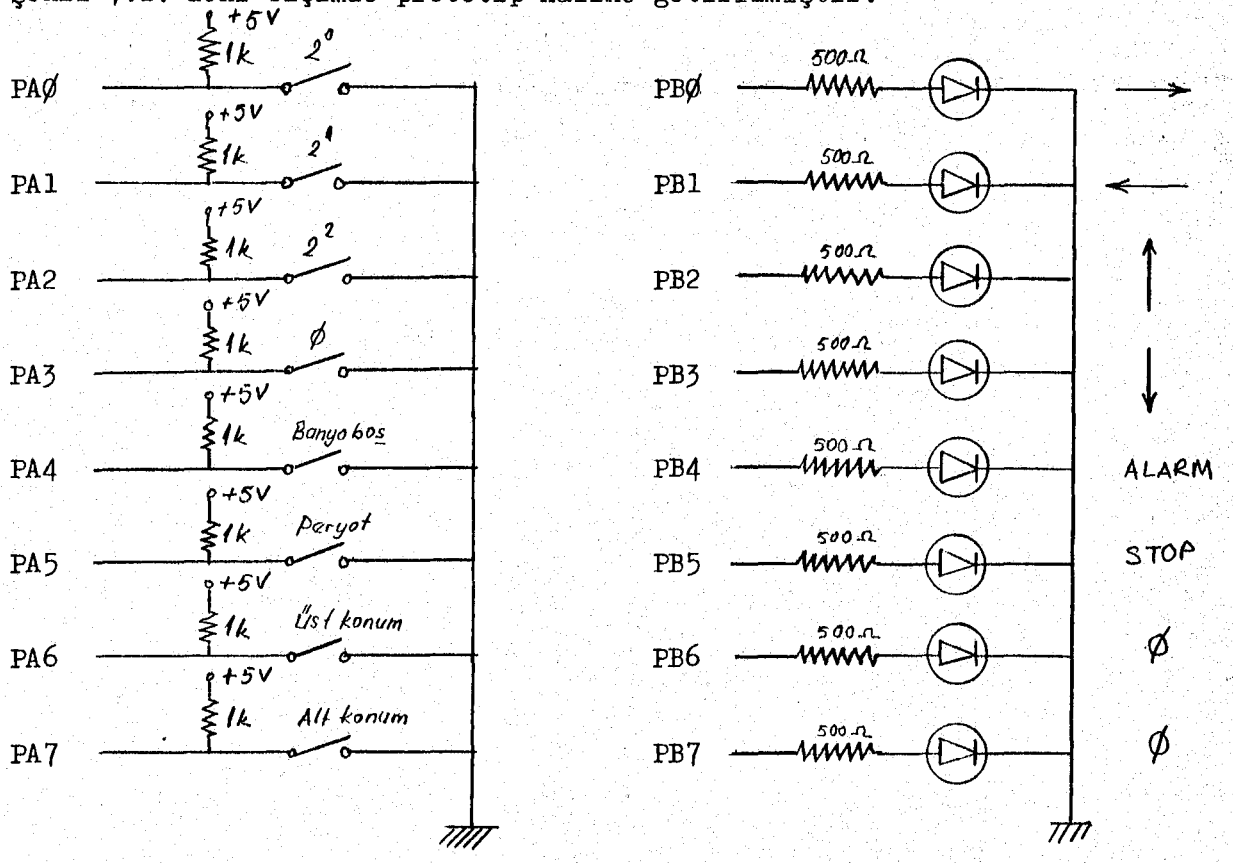
PAØ	2^0	Mikrobilgisayara giriş
PA1	2^1	
PA2	2^2	
PA3	Ø	
PA4	Banyo boş bilgisi	
PA5	Peryot	
PA6	Üst konum bilgisi	
PA7	Alt konum bilgisi	

PBØ	→	Sağa komutu	Mikrobilgisayardan Sisteme çıkış
PB1	←	Sola komutu	
PB2	↑	Yukarı komutu	
PB3	↓	Aşağı komutu	
PB4	Alarm	Alarm komutu	
PB5	Stop	Stop komutu	
PB6			
PB7			

Tablo 7.1, 2.PIA giriş/çıkış düzenlemesi.

./..

Bu tez çalışmasında kaplama hattının modellemesi Şekil 7.1. deki biçimde prototip haline getirilmiştir.



Şekil 7.1. Prototip donanımı.

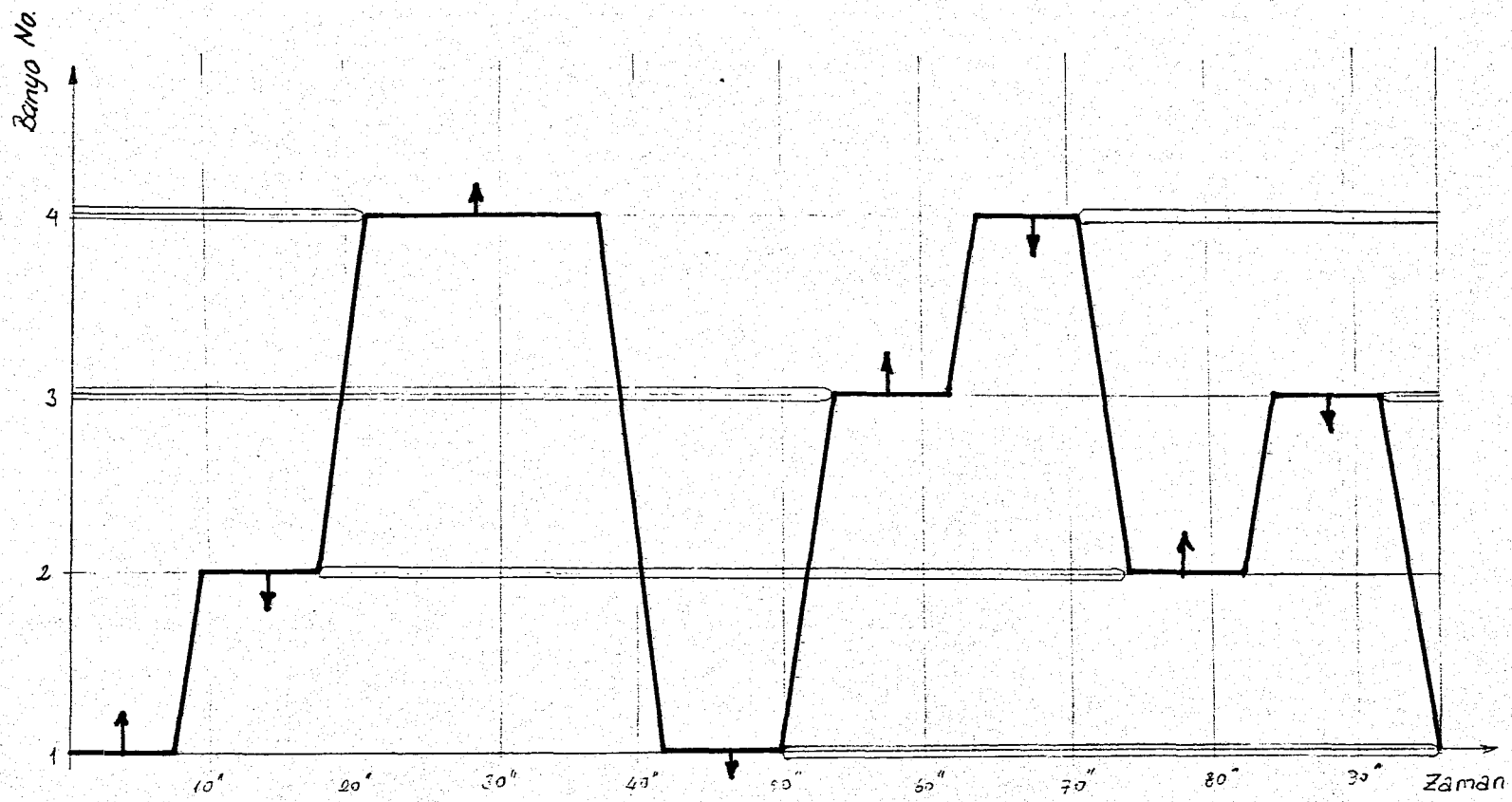
Şekil 7.1. de prototip donanımına bakıldığında, sisteme giriş bilgilerinin mikroanahtarlarla, çıkış bilgileri de LED diodlarla simule edildiği görülmektedir.

./..

7.2. Taşıyıcı hareket diagramı.

6. bölümde anlatılan kaplama hattındaki taşıyıcı hareket diagramı işlemin kimyasal özelliği ile ilgili olduğu belirtilmişti. Model olarak dört banyoluk bir kaplama hattının hareket diagramı Şekil 7.2. deki gibidir.

./..



Şekil 7.2_ Taşıyıcı hareket diagramı

Şekil 7.2. de yatay eksen zamanı, dikey eksen banyo numaralarını kalın çizgiler de taşıyıcı hareketini göstermektedir.

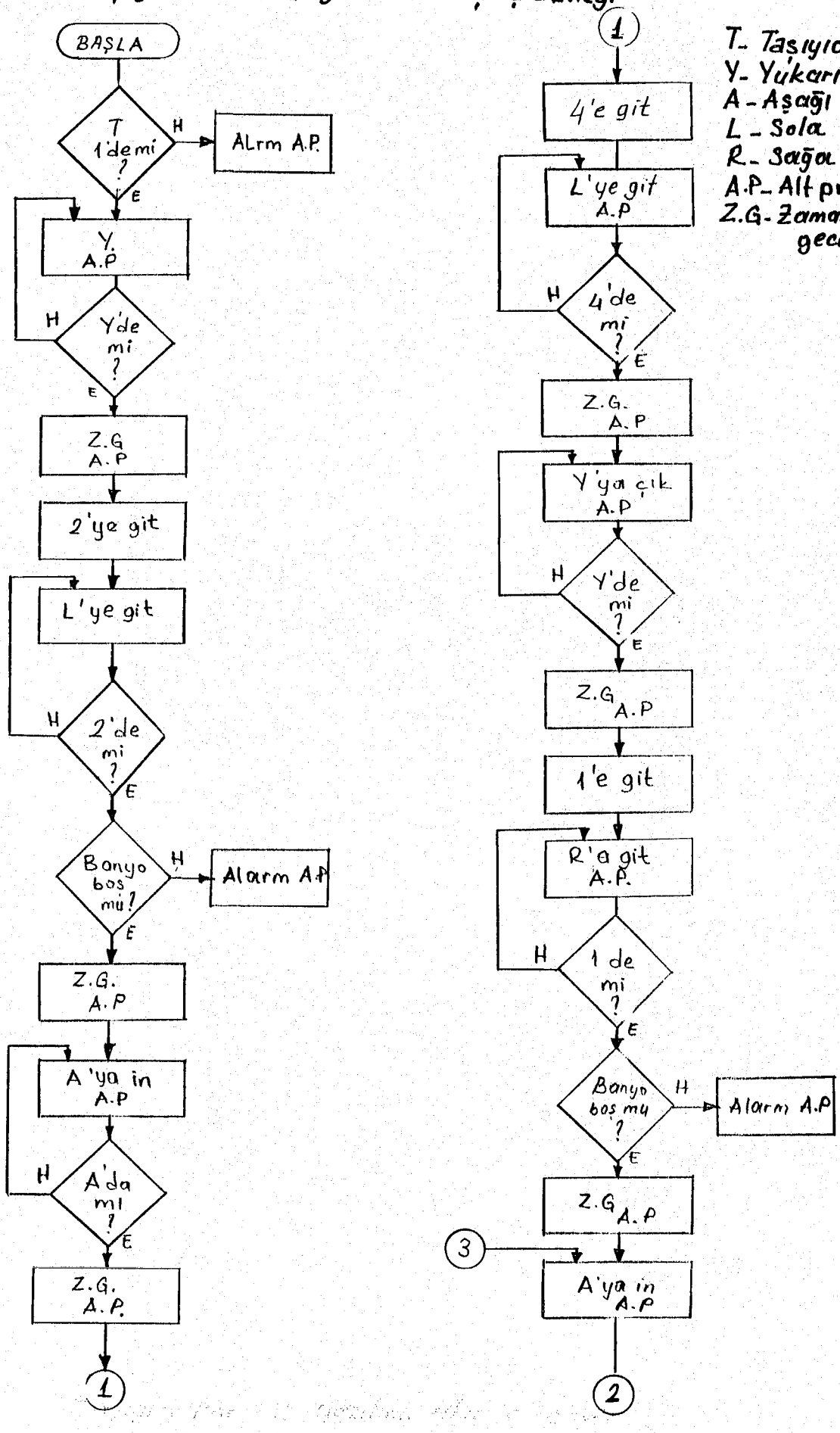
7.3. Kontrol yazılımı akış çizeneği

Kontrol edilecek taşıyıcının hareket diagramı oluşturulduktan sonra EK 1 de gösterilen yazılım akış çizeneği yazılımda kolaylıklar sağlar, programın çalışmasını daha anlaşılır kılar.

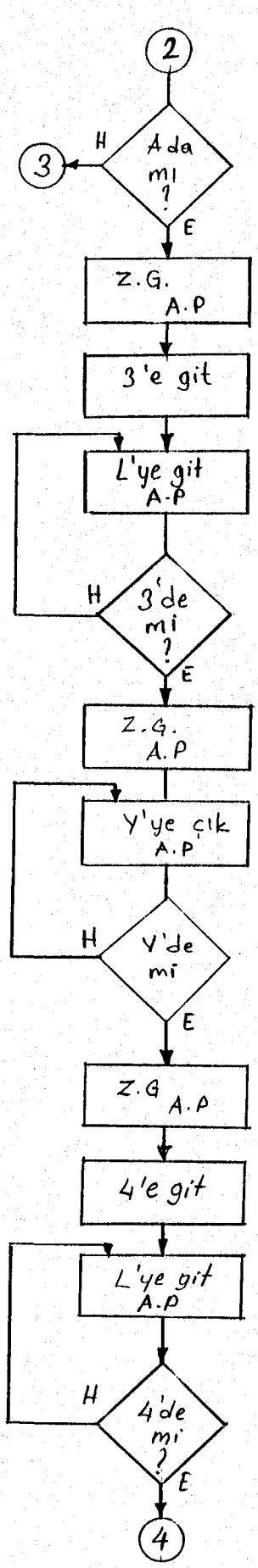
Yazılım akış çizeneği EK 1 de verilmiştir.

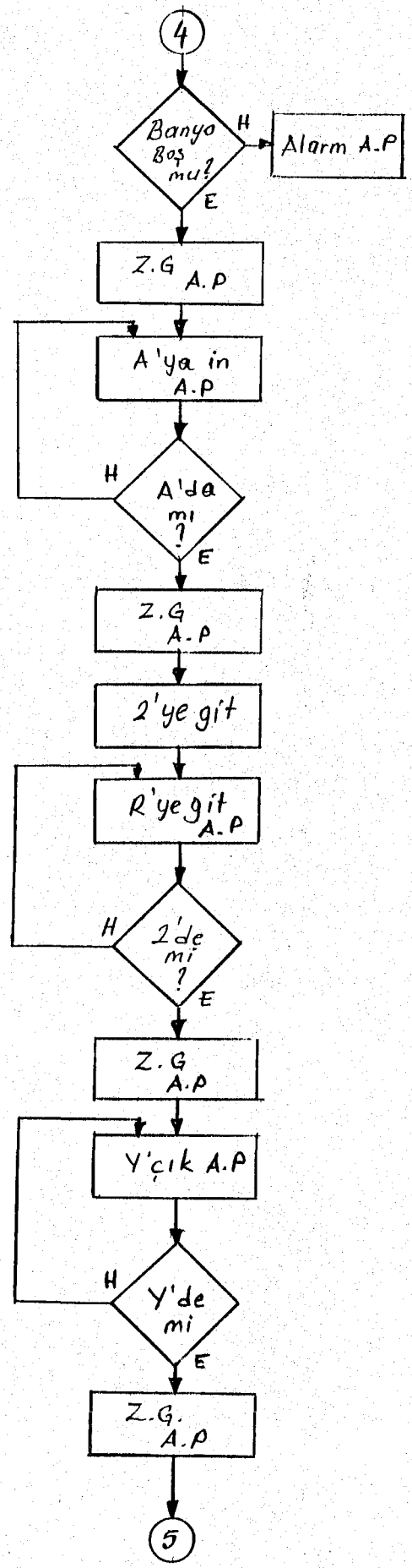
./..

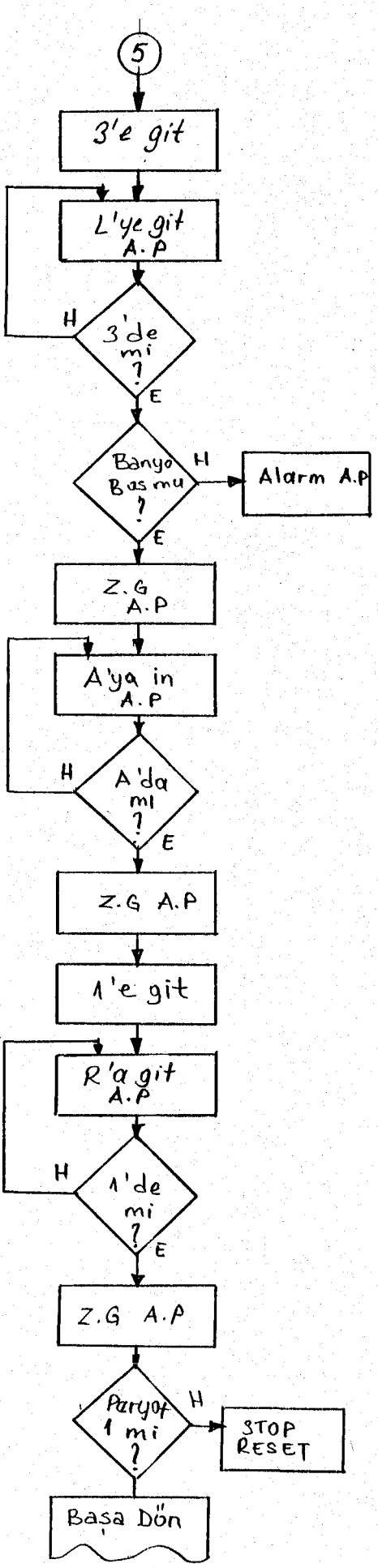
EK_1 Taşıyıcı kontrol yazılımı akış çizeneği



T- Taşıyıcı
Y- Yukarı
A - Aşağı
L - Sola
R - Sağa
A.P - Alt program
Z.G - Zaman gecikmesi


2
Ada mı ?
H
E
3
Z.G. A.P
3'e git
L'ye git A.P
3'de mi ?
H
E
Z.G. A.P
Y'ye çık A.P
Y'de mi
H
E
Z.G A.P
4'e git
L'ye git A.P
4'de mi ?
H
E
4


4
Banyo Boş mu?
H
E
Alarm A.P
Z.G A.P
A'ya in A.P
A'da mı ?
H
E
Z.G A.P
2'ye git
R'ye git A.P
2'de mi ?
H
E
Z.G A.P
Y'çık A.P
Y'de mi
H
E
Z.G. A.P
5


5
3'e git
L'ye git A.P
3'de mi ?
H
E
Banyo Bas mu ?
H
Alarm A.P
E
Z.G A.P
A'ya in A.P
A'da mi ?
H
E
Z.G A.P
1'e git
R'a git A.P
1'de mi ?
H
E
Z.G A.P
Paryo 1 mi ?
H
STOP RESET
Başa Dön

7.4. Kontrol yazılımı

Taşıyıcı hareket diagramı ve yazılımı akış çizeneği oluşturulduktan sonra, bu çizeneğin 6802 mikroişlemci makina kodunda yazılımını oluşturmak gerekliair . Kontrol yazılımı EK 2 de verilmiştir.

PROGRAM : EK-2 KONTROL YAZILIMI SAYFA 1

Nr		Sembolik Dil	Operator	Adres	Mak. Kod		AÇIKLAMA
		NOP		2000	01		
		CLR	A001	01	7F	A001	
		CLR	A003	04	7F	A003	
		LDA A	# 00	07	86	00	
		STA A	A000	09	B7	A000	
		LDA A	# FF	0C	86	FF	
		STA A	A002	0E	B7	A002	
		LDA A	# 3C	11	86	3C	
		STA A	A001	13	B7	A001	
		STA A	A003	16	B7	A003	
		NOP		19	01		
		LDA A	A000	1A	B6	A000	
		AND A	# 8F	1D	84	8F	
		STA A	27FF	1F	B7	27FF	
		LDA A	# 81	22	86	81	
		CMP A	27FF	24	81	27FF	
	Dallan 1	BEQ	- 05-	27	27	05	
		JSR	0000	29	BD	0000	S1 Alt Programına
		SWI		2C	3F		
	D1	NOP		2D	01		
		JSR	000A	2E	BD	000A	S2 Alt Programına
		NOP		31	01		
		JSR	001A	32	BD	001A	S3 Alt Programına
		NOP		35	01		
		BRA	-05-	36	20	05	
		NOP		38	01		
		JSR	002A	39	BD	002A	S4 Alt Programına
		LDA A	A000	3C	B6	A000	
		AND A	# 4F	3F	84	4F	
		STA A	27FE	41	B7	27FE	
		LDA A	# 42	44	86	42	
		CMP A	27FE	46	B1	27FE	
		BNE	- ED-	49	26	ED	
		NOP		4B	01		
		LDA A	A000	4C	B6	A000	
		ROL A		4F	49		
		ROL A		50	49		
		ROL A		51	49		
		ROL A		52	49		
		BCC	-08-	53	08		
		NOP		55	01		
		JSR	001A	56	BD	001A	
		NOP		59	01		
		BRA	-06-	5A	20	06	
		NOP		5C	01		
		JSR	0000	5D	BD	0000	
		SWI		60	3F		
		NOP		61	01		

PROGRAM :

SAYFA 2

Nr		Sembolik Dil	Operator	Adres	Mak. Kod		AÇIKLAMA
		LDA A	A000	2062	B6	A000	
		ROL A		65	49		
		BCS	-08-	66	25	08	
		LDA B	#08	68	C6	08	
		STA B	A002	6A	F7	A002	
		BRA	-F2-	6D	20	F2	
		NOP		6F	01		
		JSR	001A	70	BD	001A	
		BRA	-05-	73	20	05	
		NOP		75	01		
		JSR	002A	76	BD	002A	
		LDA A	A000	79	B6	A000	
		AND A	# 8F	7C	84	8F	
		STA A	27FD	7E	B7	27FD	
		LDA A	# 84	81	86	84	
		CMP A	27FD	83	B1	27FD	
		BNE	-ED-	86	26	ED	
		NOP		88	01		
		JSR	001A	89	BD	001A	
		NOP		8C	01		
		JSR	000A	8D	BD	000A	
		NOP		90	01		
		JSR	001A	91	BD	001A	
		NOP		94	01		
		BRA	-05-	95	20	05	
		NOP		97	01		
		JSR	003A	98	BD	003A	
		NOP		9B	01		
		LDA A	A000	9C	B6	A000	
		AND A	# 4F	9F	84	4F	
		STA A	27FC	A1	B7	27FC	
		LDA A	#41	A4	86	41	
		CMP A	27FC	A6	B1	27FC	
		BNE	-ED-	A9	26	ED	
		NOP		AB	01		
		LDA A	A000	AC	B6	A000	
		ROL A		AF	49		
		ROL A		B0	49		
		ROL A		B1	49		
		ROL A		B2	49		
		BCC	-08-	B3	24	08	
		NOP		B5	01		
		JSR	001A	B6	BD	001A	
		NOP		B9	01		
		BRA	-06-	BA	20	06	
		NOP		BC	01		
		JSR	0002	BD	BD	0000	
		SWI		C0	3F		

PROGRAM :

SAYFA 3

Nr		Sembolik Dil	Operator	Adres	Mak. Kod		AÇIKLAMA
		NOP		20C1	01		
		LDA A	A000	C2	B6	A000	
		ROL A		C5	49		
		BCS	-08-	C6	25	08	
		LDA B	#08	C8	08		
		STA B	A002	CA	F7	A002	
		BRA	-F3-	CD	20	F3	
		NOP		CF	01		
		JSR	001A	D0	BD	001A	
		NOP		D3	01		
		BRA	-05-	D4	20	05	
		NOP		D6	01		
		JSR	002A	D7	BD	002A	
		LDA A	A000	DA	B6	A000	
		AND A	# 8F	DD	84	8F	
		STA A	27FB	BF	B7	27FB	
		LDA A	# 83	E2	86	83	
		CMP A	27FB	E4	B1	27FB	
		BNE	-ED-	E7	26	ED	
		NOP		E9	01		
		JSR	001A	EA	BD	001A	
		NOP		ED	01		
		JSR	000A	EE	BD	000A	
		NOP		F1	01		
		JSR	001A	F2	BD	001A	
		NOP		F5	01		
		BRA	-05-	F6	20	05	
		NOP		F8	01		
		JSR	002A	F9	BD	002A	
		LDA A	A000	FC	B6	A000	
		AND A	# 4F	FF	84	4F	
		STA A	27FA	2101	B7	27FA	
		LDA A	#44	04	86	44	
		CMP A	27FA	06	B1	27FA	
		BNE	-ED-	09	26	ED	
		NOP		0B	01		
		LDA A	A000	0C	B6	A000	
		ROL A		0F	49		
		ROL A		10	49		
		ROL A		11	49		
		ROL A		12	49		
		BCC	-08-	13	24	08	
		NOP		15	01		
		JSR	001A	16	BD	001A	
		NOP		19	01		
		BRA	-06-	1A	20	06	
		NOP		1C	01		
		JSR	0000	1D	BD	0000	

PROGRAM :

Nr		Sembolik Dil	Operator	Adres	Mak. Kod		AÇIKLAMA
		SWI		2120	3F		
		NOP		21	01		
		LDA A	A000	22	B6	A000	
		ROL A		25	49		
		BCS	-08-	26	25	08	
		LDA B	# 08	28	C6	08	
		STA B	A002	2A	F7	A002	
		BRA	- F2 -	2D	20	F2	
		NOP		2F	01		
		JSR	001A	30	BD	001A	
		NOP		33	01		
		BRA	-05-	34	20	05	
		NOP		36	01		
		JSR	003A	37	BD	003A	
		NOP		3A	01		
		LDA A	A000	3B	B6	A000	
		AND A	# 8F	3E	84	8F	
		STA A	27F9	40	B7	27F9	
		LDA A	# 82	43	86	82	
		CMP A	27F9	45	B1	27F9	
		BNE	-EC-	47	26	EC	
		NOP		4A	01		
		JSR	001A	4B	BD	001A	
		NOP		4E	01		
		JSR	000A	4F	BD	000A	
		NOP		52	01		
		JSR	001A	53	BD	001A	
		NOP		56	01		
		BRA	-05-	57	20	05	
		NOP		59	01		
		JSR	002A	5A	BD	002A	
		LDA A	A000	5D	B6	A000	
		AND A	# 4F	60	86	4F	
		STA A	27F8	62	B7	27F8	
		LDA A	# 43	65	86	43	
		CMP A	27F8	67	B1	27F8	
		BNE	-ED-	6A	26	ED	
		NOP		6C	01		
		LDA A	A000	6D	B6	A000	
		ROL A		70	49		
		ROL A		71	49		
		ROL A		72	49		
		ROL A		73	49		
		BCC	-08-	74	24	08	
		NOP		76	01		
		JSR	001A	77	BD	001A	
		NOP		7A	01		
		BRA	-06-	7B	20	06	

PROGRAM :

SAYFA 5

Nr	Sembolik Dil	Operator	Adres	Mak. Kod		AÇIKLAMA
	NOP		217D	01		
	JSR	0000	7E	BD	0000	
	SWI		81	3F		
	NOP		82	01		
	LDA A	A000	B3	B6	A000	
	ROL A		86	49		
	BCS	-08-	87	25	08	
	LDA B	# 08	89	C6	08	
	STA B	A002	8B	F7	A002	
	BRA	-F2-	8E	20	F2	
	NOP		90	01		
	JSR	001A	91	BD	001A	
	NOP		94	01		
	BRA	-05-	95	20	05	
	NOP		96	01		
	JSR	003A	98	BD	003A	
	NOP		9B	01		
	LDA A	A000	9C	B6	A000	
	AND A	8F	9F	84	8F	
	STA A	27F7	A1	B7	27F7	
	LDA A	# 81	A4	86	81	
	CMP A	27F7	A6	B1	27F7	
	BNE	-FC-	A9	26	FC	
	NOP		AB	01		
	JSR	001A	AC	BD	001A	
	NOP		AF	01		
	LDA A	A000	B0	B6	A000	
	ROL A		B3	49		
	ROL A		B4	49		
	ROL A		B5	49		
	BCS	-08-	B6	25	08	
	LDA A	# 20	B8	86	20	
	STA A	A002	BA	B7	A002	
	BRA	-F9-	BD	20	F9	
	NOP		BF	01		
	JMP	2000	21C0	7E	2000	

ALT PROGRAMLAR

S1 Alt Programı 0000

```
0000  01       NOP
  01  86 10    LDA A #10
  03  CE 07FF  LDX #07FF
  06  09       DECX
  07  26 F7    BNE F7
  09  39       RTS
```

S2 Alt Programı 000A

```
000A  01       NOP
  0B  86 04    LDA A #04
  0D  B7 A002  STA A A002
  10  F6 A000  LDA B A000
  13  59       ROL B
  14  59       ROL B
  15  59       ROL B
  16  24 F2    BCC F2
  18  39       RTS
```

S3 Alt Programı 001A

```
001A  01       NOP
  1B  CE 01FF  LDX #01FF
  1E  09       DECX
  1F  26 F9    BNE F9
  11  39       RTS
```

S4 Alt Programı 002A

```
002A  01       NOP
  2B  86 02    LDA A #02
  2D  B7 A002  STA A A002
  30  39       RTS
```

S5 Alt Programı

```
003A  01       NOP
  3B  86 01    LDA A #01
  3D  B7 A002  STA A A002
  40  39       RTS
```

KAYNAKÇA

1- Motorola,1984,"8 Bit Microprocessors Data Book"

2- ROBİNSON,G.P.S,1983,"How to programming use 6800"

3- YUEN,C.K,1982,"Microprocessor System"

4- CERİD,Ö.1985,"Mikroişlemciler" B.Ü.Seminer notları

5- CERİD,Ö.1985,"Mikroişlemciler ve endüstriyel kontrol"B.Ü. Seminer notları

6- Mc Dermıt Co.1974,"Electroplating proces"