

## İçindekiler

1	BÖLÜM GİRİŞ .....	3
1.1	Genel Bilgisayar Yapısı .....	3
1.2	Bilgisayar Mimarisi .....	4
1.2.1	Mikroişlemcinin Çalışması .....	5
1.2.2	Mikroişlemci Tasarım Yapıları .....	6
1.3	Mikroişlemcilerin Gelişimi .....	12
1.3.1	Mikroişlemcinin Tarihsel Gelişimi .....	12
1.3.2	80X86 Ailesinin Gelişimi .....	13
1.4	8-Bitlik Mikroişlemciler: .....	15
1.5	16-Bitlik Mikroişlemciler: .....	18
1.6	32-Bitlik Mikroişlemciler: .....	19
2	BÖLÜM-2 SAYISAL LOJİK DEVRELERİN ÖZETİ .....	20
2.1	Sayısal Bilgisayar .....	20
2.2	Lojik Kapılar .....	21
2.2.1	Boole Cebri .....	22
2.2.2	DeMorgan Kuralı .....	23
2.2.3	Karnaugh Haritasının Basitleştirilmesi .....	24
2.3	Kombinasyonel Devreler .....	25
2.3.1	Yarı-Toplayıcı (HA:Half Adder) .....	25
2.4	Flip-Floplar (FF).....	28
2.4.1	Kenar Tetiklemeli ve Efendi-Köle (Master-Slave) Tipi FF'ler .....	29
2.5	Ardışıl Devreler .....	30
3	BÖLÜM SAYISAL BİLEŞENLER .....	33
3.1	Tümdevrelerin Tanıtımı.....	33
3.2	Dekoder-Enkoder.....	34
3.2.1	Dekoder .....	34
3.2.2	Enkoder .....	35
3.3	Çoğullayıcı (Mux) .....	36
3.4	Bellek Elemanı (Register) Saklayıcı.....	37
3.5	Kaydırmalı Bellek Elemanı .....	39
3.6	Bit İki Yönlü Kaydırmalı Paralel Yükleme Bellek Elemanı .....	40
3.7	İkili Sayıcı.....	41
3.8	Ana Bellek Ünitesi .....	41
3.9	Olasıl (Rastgele) Erişimli Bellek (RAM) .....	41
3.10	Yalnızca Okunabilir Bellek (ROM).....	42
4	BÖLÜM YAZGAÇ –SAKLAYICI (Register) TRANSFER DİLİ ve MİKROİŞLEMLER	44
4.1	Saklayıcı (Register) Transfer Dili .....	44
4.1.1	R1 den R2 ye Bilgi Transferi .....	44
4.1.2	R1 den R2 ye Koşullu Bilgi Transferi.....	45
4.2	Hat Grubu (Bus) Yoluyla Saklayıcılar Arası Veri Transferi.....	45
4.2.1	MUX (Multiplexer Bus) Yoluyla Genel Hat Grubu (BUS) Bağlantısının Yapılması .....	46
4.2.2	Üç-Durumlu Kapılar Yoluyla Genel Hat Grubu (BUS) Bağlantısının Yapılması	46
4.2.3	Ana Bellek Bilgi Transferi .....	48
4.3	Aritmetik Mikroişlemler.....	49
4.3.1	2-Bitlik Toplama-Çıkarma İşlemi.....	49
4.3.2	2-Bitlik Arttırma İşlemi .....	51

4.3.3	1-Bitlik Aritmetik Devre Tasarımı .....	51
4.3.4	Aritmetik Mikroişlemciler Tablosunda Aritmetik Devrenin Gerçekleştirdiği İşlemler .....	52
4.4	1-bitlik Lojik Mikroişlem Biriminin Gerçeklenmesi.....	55
4.5	Kaydırma (Shift ve Rotate) Mikroişlemleri .....	56
4.5.1	Lojik Kaydırma .....	57
4.5.2	Dairesel Kaydırma.....	57
4.5.3	Aritmetik Kaydırma .....	57
4.5.4	Kaydırma Yapan Saklayıcının Tasarımı .....	58
4.6	Aritmetik, Lojik, Kaydırma (Alu) Birimi.....	60
5	TEMEL SAYISAL BİLGİSAYARIN KOMUT ve SAKLAYICI (REGISTER) YAPISI ..	62
5.1	Bilgisayarın Adresleme Modları .....	63
5.2	Bilgisayarın Saklayıcı Yapısı .....	64
5.3	Bilgisayarın Bus Yapısı .....	66
5.4	Komut Yapısı.....	70
5.4.1	Saklayıcı Adreslemeli Komut Listesi.....	70
5.4.2	Giriş/Çıkış Adreslemeli Komut Listesi .....	71
5.5	Tam Komut Seti ve Verimliliği .....	71
5.6	Zamanlama ve Kontrol .....	72
5.6.1	Kontrol Birimi .....	73
5.6.2	Zamanlama Birimi.....	74
5.7	Komut Fazları .....	75
5.7.1	Komutun alınması .....	76
5.8	Komut Yapıları .....	77
5.8.1	Komut Tipinin Belirlenmesi ve Dallanma .....	77
5.8.2	Saklayıcı Adreslemeli Komutlar .....	78
5.8.3	Ana Bellek Adreslemeli Komutlar .....	79
5.9	Giriş/Çıkış Ve Kesinti Kavramı .....	82
5.9.1	Kesinti Programı ile Veri Giriş/Çıkışı.....	83
5.9.2	Güncellenmiş Komutun Alınması Fazı .....	86
5.10	Temel Bilgisayarın Tasarımı .....	87
5.11	Kontrol Mantık Kapıları .....	88
5.11.1	Zamanlama Kontrol İşareti: .....	92
5.11.2	AR Saklayıcı: .....	93
5.11.3	DR Saklayıcı: .....	96
5.11.4	IR Saklayıcı: .....	97
5.11.5	TR Saklayıcı: .....	98
5.11.6	OUTR Saklayıcı: .....	99
5.11.7	INPR Saklayıcı: .....	99
5.11.8	Memory Read, Memory Write: .....	99
5.11.9	AC Saklayıcı: .....	100
5.11.10	Bayraklar .....	102
5.12	BUS KONTROL.....	106
6	TEMEL BİLGİSAYARIN PROGRAMLAMASI .....	111
6.1	Temel Bilgisayarın Assembler Komutları .....	111
6.2	İkili Kod ve Assembler Kodu Arasındaki İlişki .....	111
6.3	Karşılaştırmalı Program Örneği .....	111
6.4	Assembly Dili Detayları .....	112
6.5	Örnek Uygulamalar.....	113

# 1 BÖLÜM GİRİŞ

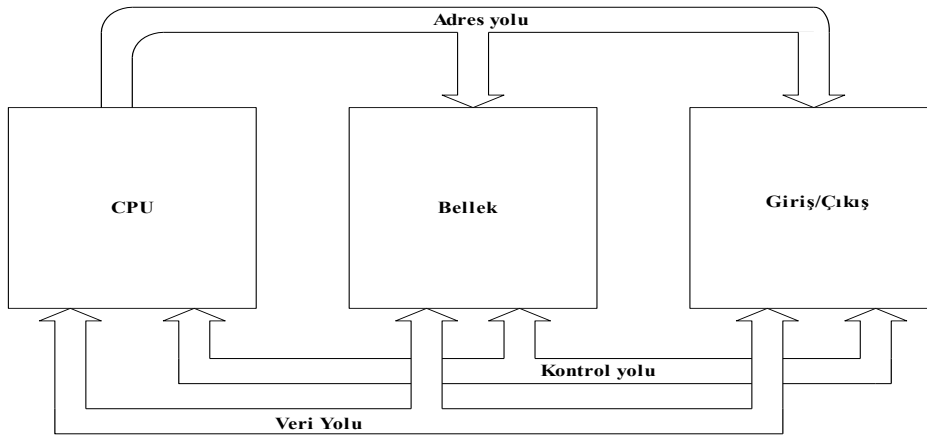
Mikroişlemci, saklı bir komut dizisini ardışıl olarak yerine getirerek veri kabul edebilen ve bunları işleyebilen sayısal bir elektronik eleman olarak tanımlanabilir. **Mikroişlemci temelde mantık kapıları, flip-floplar, sayıcı ve saklayıcılar gibi standart sayısal devrelerden oluşur.**

Genel olarak bilgisayar ile iki şekilde ilgilenilir :

**1. Yazılım (Software) : Bilgisayarın fiziksel parçalarını işler hale getiren bileşenlerdir.**

**2. Donanım (Hardware) : Bilgisayarı oluşturan fiziksel parçaların tümüdür.**

Her ikisi de birbirinin tamamlayıcısıdır. Birisi olmazsa diğeri de olmaz. Sistem öncelikli olarak tasarlanırken önce sistemi meydana getirecek elemanlar, yani donanım parçaları göz önüne alınır. Daha sonra yazılım bu yapıya bakılarak yazılır. Yazılım, donanımın hangi yönetime göre nasıl çalışacağını gösteren bir sanal uygulamadır. Hangi zamanda hangi elemanın devreye girerek üzerindeki bilgiyi işlemesini sağlamaktadır. Basit bir bilgisayarın ana elemanları **Şekil 1.'de** görülmektedir. Tüm sayısal bilgisayarlar şekilde gösterilen elemanlara sahiptirler. Bunların dışındaki eleman ya da cihazlar seçimlidir.



## 1.1 Genel Bilgisayar Yapısı

Bilgisayarı oluşturan bu sistemdeki elemanlar; mikroişlemci(CPU), bellek ve giriş/çıkış(G/Ç) birimleridir. **Mikroişlemcinin işleyeceği komutlar ve veriler geçici veya kalıcı belleklerde tutulmaktadır. Bilgiyi oluşturan komut ve veriler bellekte karmaşık veya farklı alanlarda tutulabilir.**

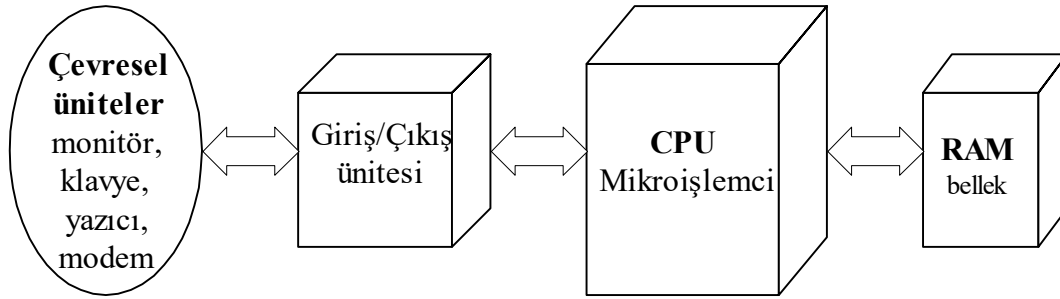
CPU tarafından gerçekleştirilen iki temel işlem vardır. Birincisi komutların yorumlanarak doğru bir sırada gerçekleşmesini sağlayan kontrol işlevi, diğeri toplama,

çıkarma vb özel matematik ve mantık işlemlerinin gerçekleştirilmesini sağlayan icra işlevidir.

## 1.2 Bilgisayar Mimarisi

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

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



Bir mikroişlemci sisteminin temel bileşenleri

*Giriş/Çıkış(Input/Output):* Sayısal, analog ve özel fonksiyonlardan oluşur ve mikroişlemcinin dış dünya ile haberleşmesini sağlar.

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

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

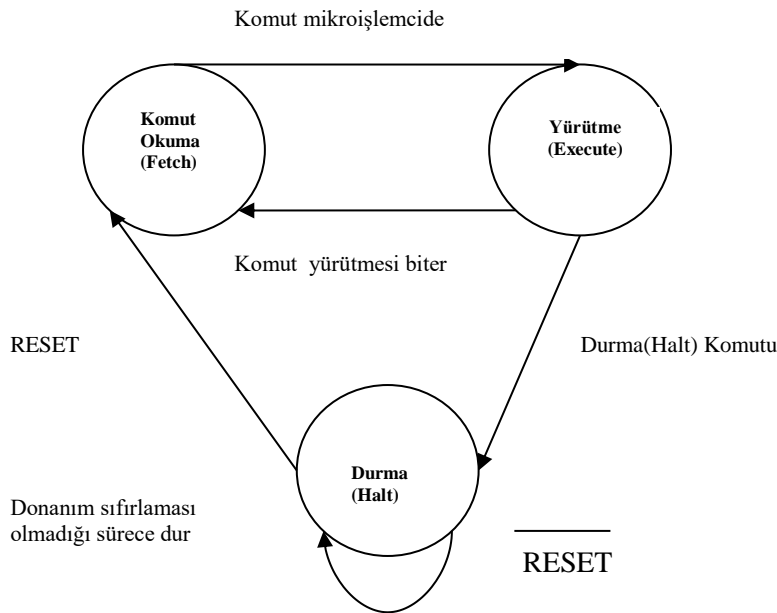
*Osilatör :* Mikroişlemcinin düzgün çalışabilmesi için gerekli olan elemanlardan biridir. Görevi; veri ve komutların CPU 'ya alınmasında, yürütülmesinde, kayıt edilmesinde, sonuçların hesaplanmasında ve çıktıların ilgili birimlere gönderilmesinde gerekli olan saat darbelerini üretmektir. Osilatör, farklı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

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

### 1.2.1 Mikroişlemcinin Çalışması

Bir mikroişlemcinin çalışmasında ,kontrol birimi tarafından yerine getirilen ve **Şekil 3** de gösterilen temel iki işlem vardır. **Komut okuma (fetch)** ve **komut yürütme (execute)**. Komut okuma,mikroişlemcinin hafızadan bir **işlem kodu (operation code-opcode)** alıp **komut saklayıcısına (Instruction Register –IR)** getirme işlemine denir. Komut saklayıcısına gelen komut ile hangi işlemin yapılacağı komut kod çözücüsü tarafından belirlenir.Gereken sinyalleme kontrol birimi tarafından üretilir.Eğer komut ile belirlenen işlem için,bazı **işlem verisine (operand)** gerek var ise,bu veriler hafızadan okunur.

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



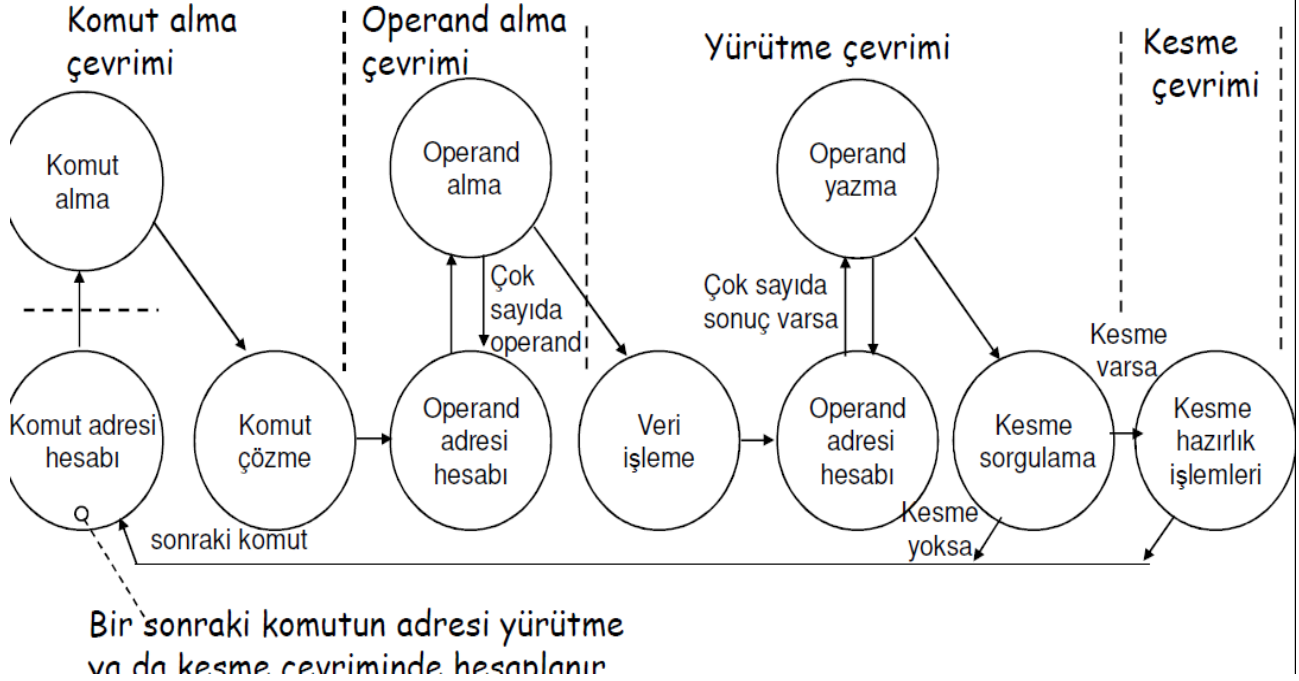
Bir mikroişlemci içindeki komut okuma ve yürütme çevrimleri

**Örnek:** Bir mikroişlemcinin program hafızasında bulunan programın çalıştırılması

### Bir Merkezi İşlem Biriminin Komut Çevrimi (Instruction Cycle):

Sonlu durumlu ardışıl makine olarak tasarlanan merkezi işlem birimi dört durumdan birinde bulunur:

1. Komut alma çevrimi, 2. Operand alma çev., 3. Komut yürütme çev., 4. Kesme çev.  
Bir merkezi işlem biriminin durum diyagramı aşağıda gösterilmiştir.



#### 1.2.2 Mikroişlemci Tasarım Yapıları

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

Mikroişlemcileri çeşitli özelliklerine göre sınıflandırmak mümkündür.

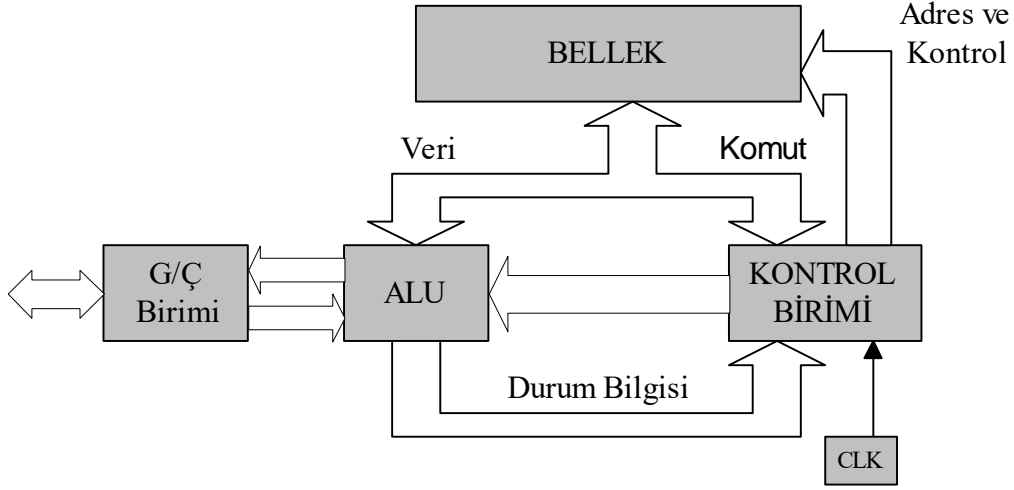
##### A- Operand Sayısına Göre

- **Sıfır operandlı mikroişlemciler.** Yığın yapıli mikroişlemciler olarakda adlandırılır.
- **Bir adresli mikroişlemciler.** Akümülatör makineleridir.
- **İki operandlı mikroişlemciler.** (Saklayıcı-Saklayıcı, Saklayıcı-Bellek, Bellek-Bellek) adresleme yapan t mikroişlemciler.
- **Üç operandlı mikroişlemciler.**

##### B- Komut ve Bellek Yapılarına Göre.

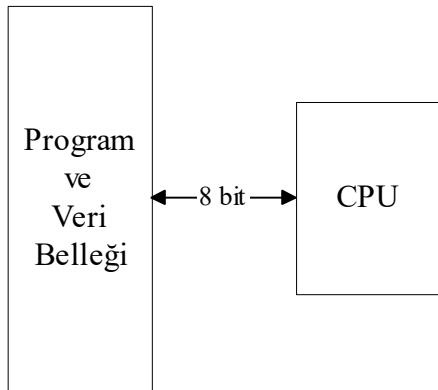
- **Von Neuman (Princeton) Mimarisi:** Bilgisayarlarda ilk kullanılan mimaridir. İlk bilgisayarlar Von Neuman yapısından yola çıkılarak geliştirilmiştir. Geliştirilen bu

bilgisayar beş birimden oluşmaktaydı. Bu birimler; aritmetik ve mantıksal birim, kontrol birim, bellek, giriş-çıkış birimi ve bu birimler arasında iletişimi sağlayan yollardan oluşur.



**Şekil 5.** Von Neuman mimarili bilgisayar sistemi

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



Von Neuman mimarisi

Von Neuman mimarisine sahip bir bilgisayar aşağıdaki sıralı adımları gerçekleştirir.

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

### Örnek 3.1:

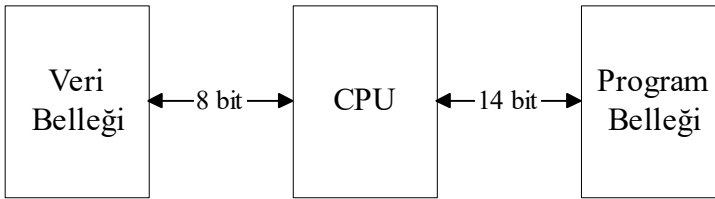
Mov acc, reg

1. cp : Komut okur

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

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

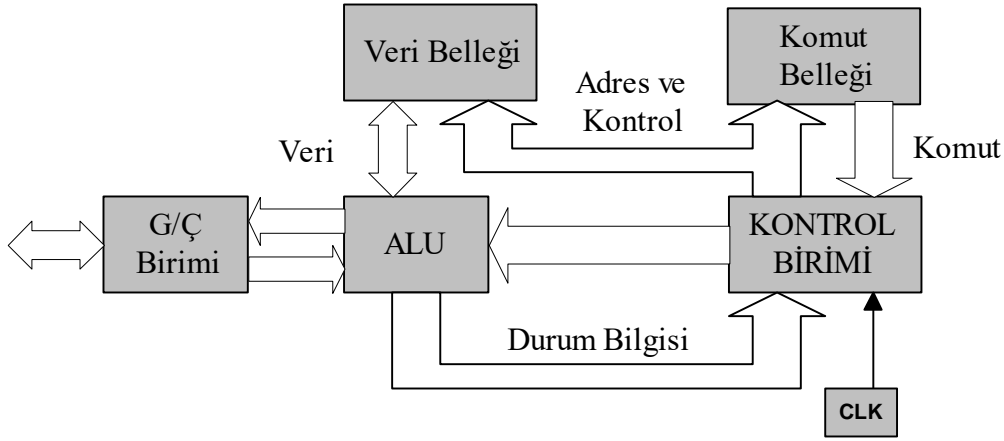
- **Harvard Mimarisi:** Harvard mimarili bilgisayar sistemlerinin Von Neuman mimarisinden farkı veri ve komutların ayrı ayrı belleklerde tutulmasıdır. Buna göre, veri ve komut aktarımında iletişim yolları da bir birinden bağımsız yapıda bulunmaktadır.



#### Harvard Mimarisi

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





Harvard Mimarili bilgisayar sistemi

### Örnek 3.2:

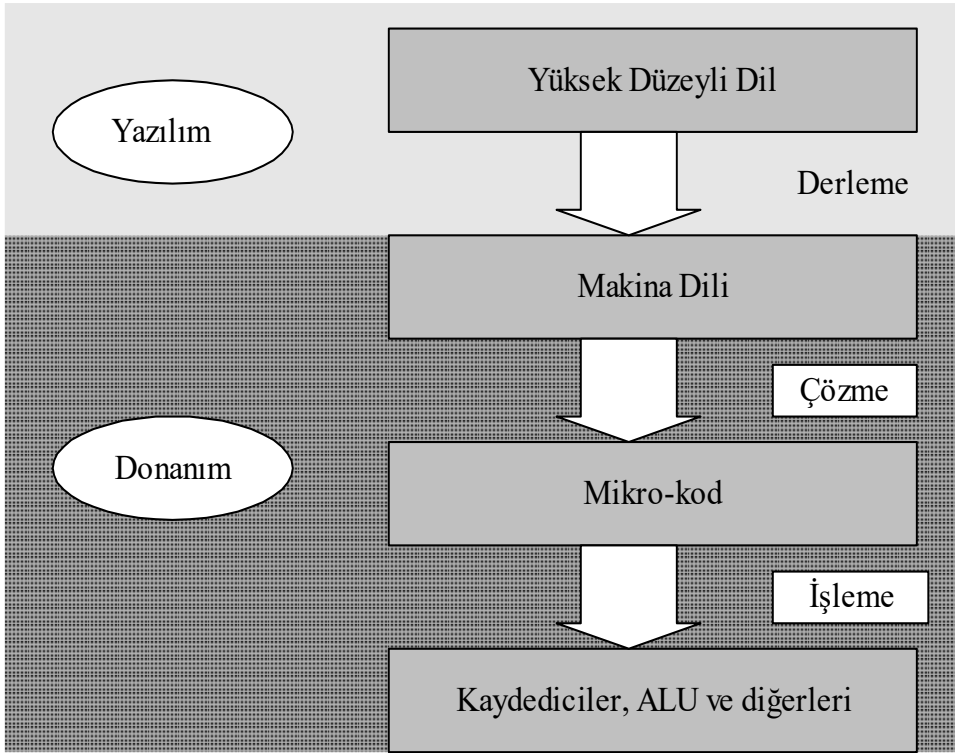
Mov acc, reg

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

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

### C- Komut Yapılarına Göre.

- **CISC (Complex Instruction Set Computer) Mimarisi :** Bu mimari, programlanması kolay ve etkin bellek kullanımı sağlayan tasarım felsefesinin bir ürünüdür. **İşlemci üzerinde performans düşüklüğü ve işlemcinin karmaşık bir hale gelmesine neden olsa da yazılımı basitleştirmektedir. Bu mimarinin en önemli iki özelliği, değişken uzunluktaki komutlar diğeri ise karmaşık komutlardır.** Karmaşık komutlar birden fazla komutu tek bir hale getirirler. Karmaşık komutlar aynı zamanda karmaşık bir mimariyi de oluşturur. Mimarideki karışıklık işlemcinin performansını da doğrudan etkilemektedir. CISC komut seti mümkün olabilen her durum için bir komut içermektedir.

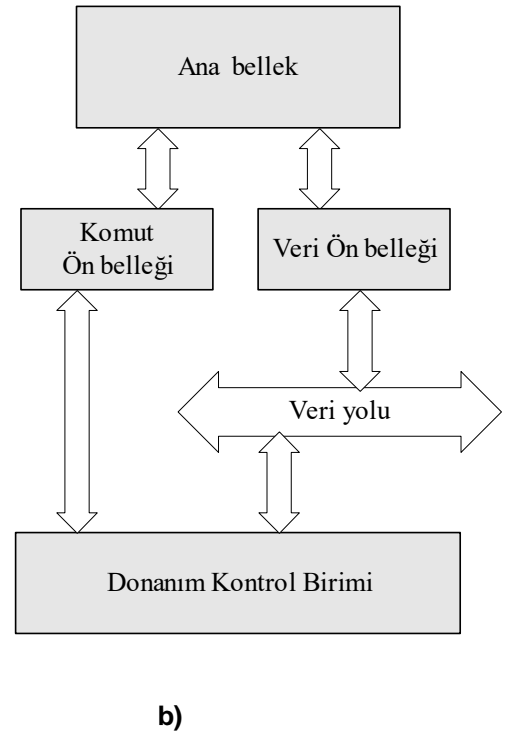
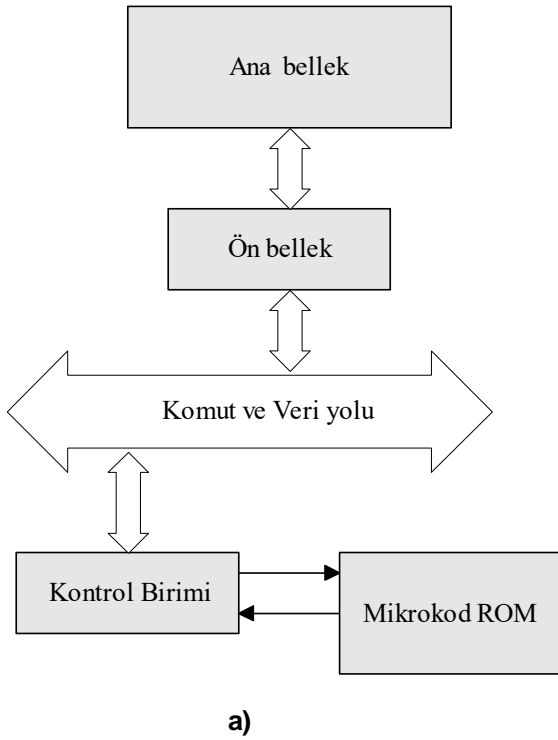


#### CISC tabanlı bir işlemcinin çalışma biçimi

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

- **RISC ( Reduced Instruction Set Computer) Mimarisi:** RISC mimarisi IBM, Apple ve Motorola gibi firmalarca sistematik bir şekilde geliştirilmiştir. RISC mimarisinin taraftarları, bilgisayar mimarisinin gittikçe daha karmaşık hale geldiğini ve hepsinin bir kenara bırakılıp en başta yeniden başlamak fikrindeydiler. 70’li yılların başında IBM firması ilk RISC mimarisini tanımlayan şirket oldu. Bu mimaride bellek hızı arttığından ve yüksek seviyeli diller assembly dilinin yerini aldığından, CISC’in başlıca üstünlükleri geçersiz olmaya başladı. RISC’in felsefesi üç temel prensibe dayanır.
- **Bütün komutlar tek bir çevrimde çalıştırılmalıdır:** Her bir komutun farklı çevrimde çalışması işlemci performansını etkileyen en önemli nedenlerden biridir. Komutların tek bir çevrimde performans eşitliğini sağlar.

- **Belleğe sadece “load” ve “store” komutlarıyla erişilmelidir.** Eğer bir komut direkt olarak belleği kendi amacı doğrultusunda yönlendirilirse onu çalıştırmak için birçok saykıl geçer. Komut alınıp getirilir ve bellek gözden geçirilir. RISC işlemcisiyle, belleğe yerleşmiş veri bir kaydediciye yüklenir, kaydedici gözden geçirilir ve son olarak kaydedicinin içeriği ana belleğe yazılır.
- **Bütün icra birimleri mikrokod kullanmadan donanımdan çalıştırılmalıdır.** Mikrokod kullanımı, dizi ve benzeri verileri yüklemek için çok sayıda çevrim demektir. Bu yüzden tek çevirimli icra birimlerinin yürütülmesinde kolay kullanılmaz.



a) Mikrokod denetimli CISC mimarisi; b) Donanım denetimli RISC mimarisi

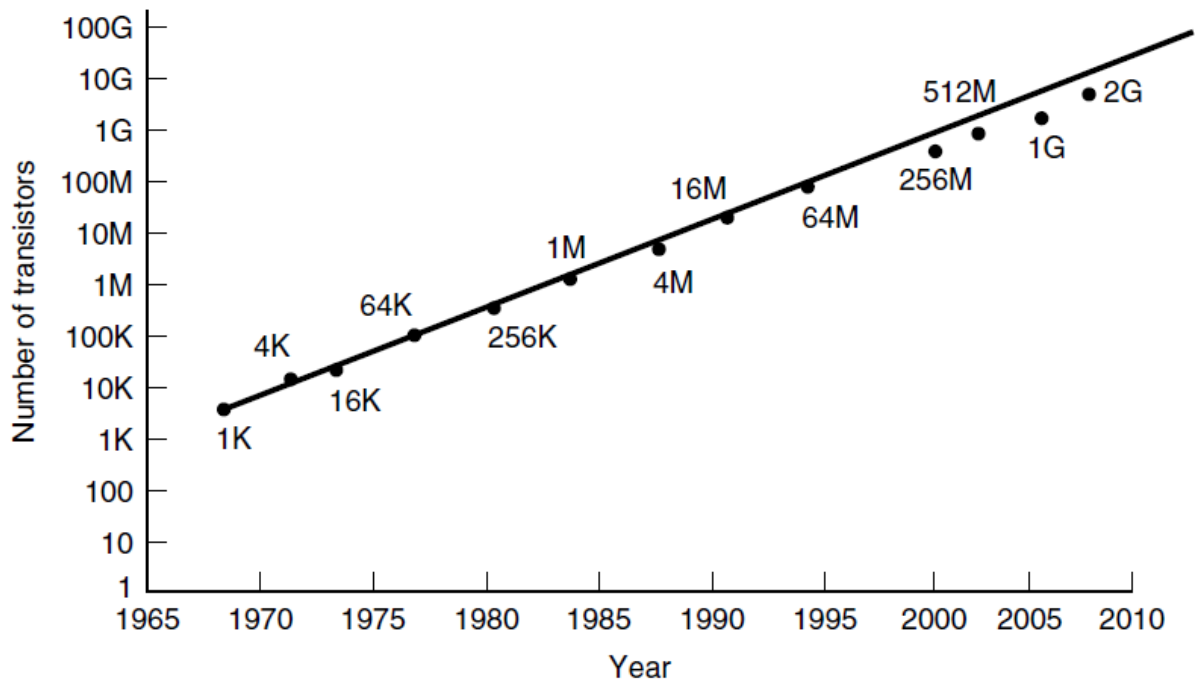
### 1.3 Mikroişlemcilerin Gelişimi

#### 1.3.1 Mikroişlemcinin Tarihsel Gelişimi

Year	Name	Made by	Comments
1834	Analytical Engine	Babbage	First attempt to build a digital computer
1936	Z1	Zuse	First working relay calculating machine
1943	COLOSSUS	British gov't	First electronic computer
1944	Mark I	Aiken	First American general-purpose computer
1946	ENIAC	Eckert/Mauchley	Modern computer history starts here
1949	EDSAC	Wilkes	First stored-program computer
1951	Whirlwind I	M.I.T.	First real-time computer
1952	IAS	Von Neumann	Most current machines use this design
1960	PDP-1	DEC	First minicomputer (50 sold)
1961	1401	IBM	Enormously popular small business machine
1962	7094	IBM	Dominated scientific computing in the early 1960s
1963	B5000	Burroughs	First machine designed for a high-level language
1964	360	IBM	First product line designed as a family
1964	6600	CDC	First scientific supercomputer
1965	PDP-8	DEC	First mass-market minicomputer (50,000 sold)
1970	PDP-11	DEC	Dominated minicomputers in the 1970s
1974	8080	Intel	First general-purpose 8-bit computer on a chip
1974	CRAY-1	Cray	First vector supercomputer
1978	VAX	DEC	First 32-bit superminicomputer
1981	IBM PC	IBM	Started the modern personal computer era
1981	Osborne-1	Osborne	First portable computer
1983	Lisa	Apple	First personal computer with a GUI
1985	386	Intel	First 32-bit ancestor of the Pentium line
1985	MIPS	MIPS	First commercial RISC machine
1985	XC2064	Xilinx	First field-programmable gate array (FPGA)
1987	SPARC	Sun	First SPARC-based RISC workstation
1989	GridPad	Grid Systems	First commercial tablet computer
1990	RS6000	IBM	First superscalar machine
1992	Alpha	DEC	First 64-bit personal computer
1992	Simon	IBM	First smartphone
1993	Newton	Apple	First palmtop computer (PDA)
2001	POWER4	IBM	First dual-core chip multiprocessor

#### Mikroişlemcilerin Tarihsel Gelişimi

**Moore kuralı:**

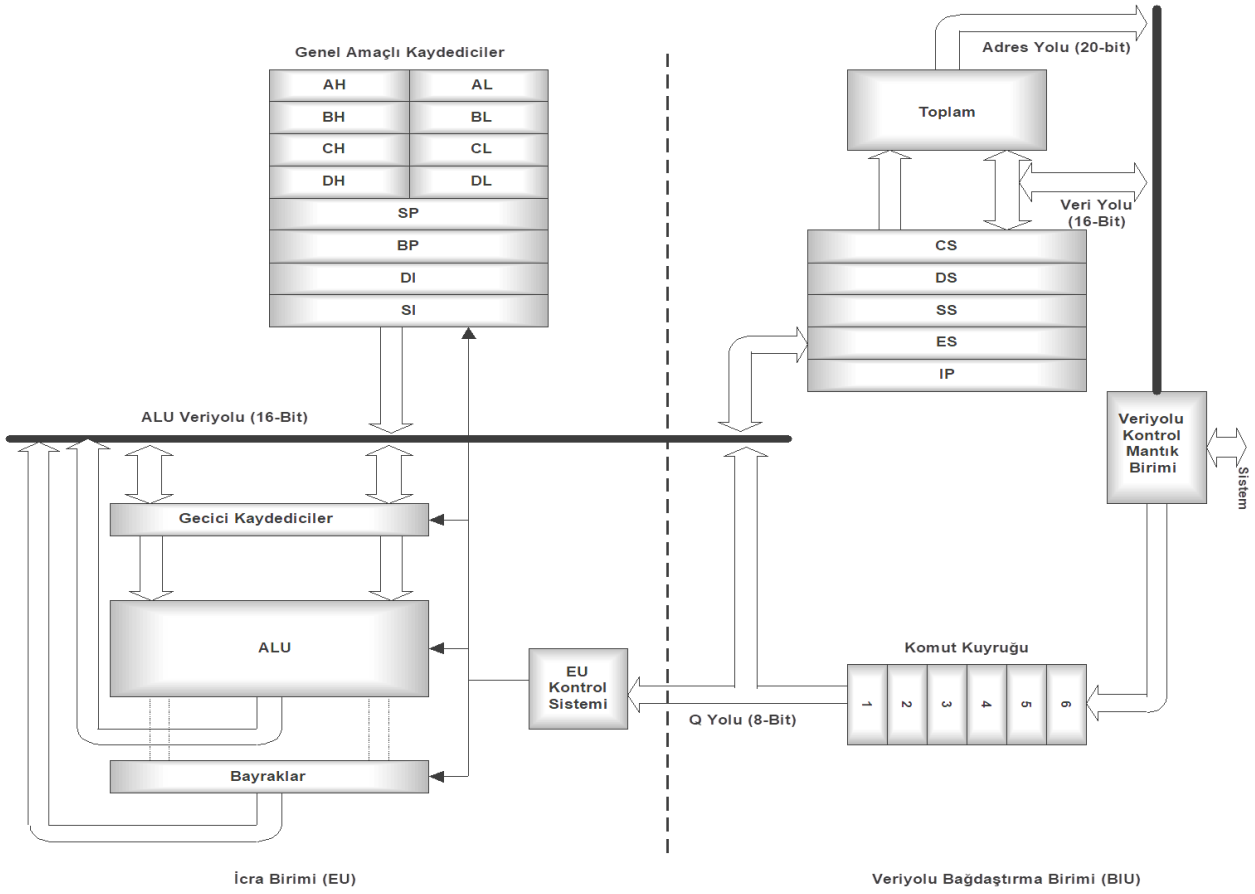


## Moore Kurah

### 1.3.2 80X86 Ailesinin Gelişimi

Chip	Date	MHz	Trans.	Memory	Notes
4004	4/1971	0.108	2300	640	First microprocessor on a chip
8008	4/1972	0.108	3500	16 KB	First 8-bit microprocessor
8080	4/1974	2	6000	64 KB	First general-purpose CPU on a chip
8086	6/1978	5–10	29,000	1 MB	First 16-bit CPU on a chip
8088	6/1979	5–8	29,000	1 MB	Used in IBM PC
80286	2/1982	8–12	134,000	16 MB	Memory protection present
80386	10/1985	16–33	275,000	4 GB	First 32-bit CPU
80486	4/1989	25–100	1.2M	4 GB	Built-in 8-KB cache memory
Pentium	3/1993	60–233	3.1M	4 GB	Two pipelines; later models had MMX
Pentium Pro	3/1995	150–200	5.5M	4 GB	Two levels of cache built in
Pentium II	5/1997	233–450	7.5M	4 GB	Pentium Pro plus MMX instructions
Pentium III	2/1999	650–1400	9.5M	4 GB	SSE Instructions for 3D graphics
Pentium 4	11/2000	1300–3800	42M	4 GB	Hyperthreading; more SSE instructions
Core Duo	1/2006	1600–3200	152M	2 GB	Dual cores on a single die
Core	7/2006	1200–3200	410M	64 GB	64-bit quad core architecture
Core i7	1/2011	1100–3300	1160M	24 GB	Integrated graphics processor

İşlemci	Çekirdek	İşlemci(Ali cron)	Clock	Çalışma Gerilimi	Registers Büyüklüğü	Data Bus Büyüklüğü	Maksimum Hafıza	L1 Cache Büyüklüğü	L2 Cache Büyüklüğü	L3 Cache Büyüklüğü	L2/L3 Cache Hızı	Multimedia Komut Kümesi	Transistors Sayısı	İlk üretim tarihi
8086	1	3,0	1x	5V	16-bit	16-bit	1MB	-	-	-	-	-	29.000	1978
386SX	1	1.5, 1.0	1x	5V	32-bit	16-bit	16MB	-	-	-	Bus Hızı	-	275.000	1978
386DX	1	1.5, 1.0	1x	5V	32-bit	32-bit	4GB	-	-	-	Bus Hızı	-	275.000	1985
486SX	1	1.0, 0.8	1x	5V	32-bit	32-bit	4GB	8KB	-	-	Bus Hızı	-	1.185M	1991
486DX	1	1.0, 0.8	1x	5V	32-bit	32-bit	4GB	8KB	-	-	Bus Hızı	-	1.2M	1989
Pentium MMX	1	0.35, 0.25	1.5x+	2.8V	32-bit	64-bit	4GB	2x16KB	-	-	Bus Hızı	MMX	4.5M	1997
Pentium Pro	1	0.35	2x+	3.3V	32-bit	64-bit	64GB	2x8KB	256KB, 512KB	-	Çekirdek Hızı	-	5.5M	1995
Pentium II (Klamath)	1	0.35	>2x	2.8V	32-bit	64-bit	64GB	2x16KB	512KB	-	1/2 Çekirdek Hızı	MMX	75M	1997
Celeron A (Mendocino)	1	0.25	>2x	2V	32-bit	64-bit	64GB	2x16KB	128KB	-	Çekirdek Hızı	MMX	19M	1998
Celeron III (Coppermine)	1	0.18	>2x	1.75V	32-bit	64-bit	64GB	2x16KB	128KB	-	Çekirdek Hızı	SSE	28.1M	2000
Celeron 4 (Willamette)	1	0.18	>2x	1.75V	32-bit	64-bit	64GB	2x16KB	128KB	-	Çekirdek Hızı	SSE2	42M	2002
Pentium 4E (Prescott)	1	0.09	>2x	1.4V	32-bit	64-bit	64GB	>16KB	1 MB	-	Çekirdek Hızı	SSE3	125M	2004
Core Duo (Yonah)	2	0.09	>2x	1.3V	32-bit	64-bit	64GB	>64KB	1MB Çekirdek başına	-	Çekirdek Hızı	SSE3	151M	2006
Core 2 Duo (Conroe)	2	0.065	>2x	1.3V	64-bit	64-bit	1TB	>64KB	2-4MB Çekirdek başına	-	Çekirdek Hızı	SSSE3	291M	2006
Atom	1	0,045	>2x	0.9V	64-bit	64-bit	2GB	32KB•	512KB	N/A	Çekirdek Hızı	SSE3, SSSE3	47M	2008
Core i7 (Ivy Bridge)	4	0,022	>2x	1.4V	64-bit	64-bit	32GB	>64KB	256KB Çekirdek başına	8MB	Çekirdek Hızı	SSE4.2, AVX	1,4B	2012
Core i5 (Ivy Bridge)	4	0,022	>2x	1.4V	64-bit	64-bit	32GB	>64KB	256KB Çekirdek başına	6MB	Çekirdek Hızı	SSE4.2, AVX	1,4	2012
Core i3/i5 (Ivy Bridge)	2	0,022	>2x	1.4V	64-bit	64-bit	32GB	>64KB	256KB Çekirdek başına	3MB	Çekirdek Hızı	SSE4.2, AVX	••••	2012
Core i7 (Haswell E)	6-8	0,022	>2x	1.4V	64-bit	64-bit	64GB	>64KB	256KB Çekirdek başına	15-20MB	Çekirdek Hızı	SSE4.2, AVX	2,6B	2013



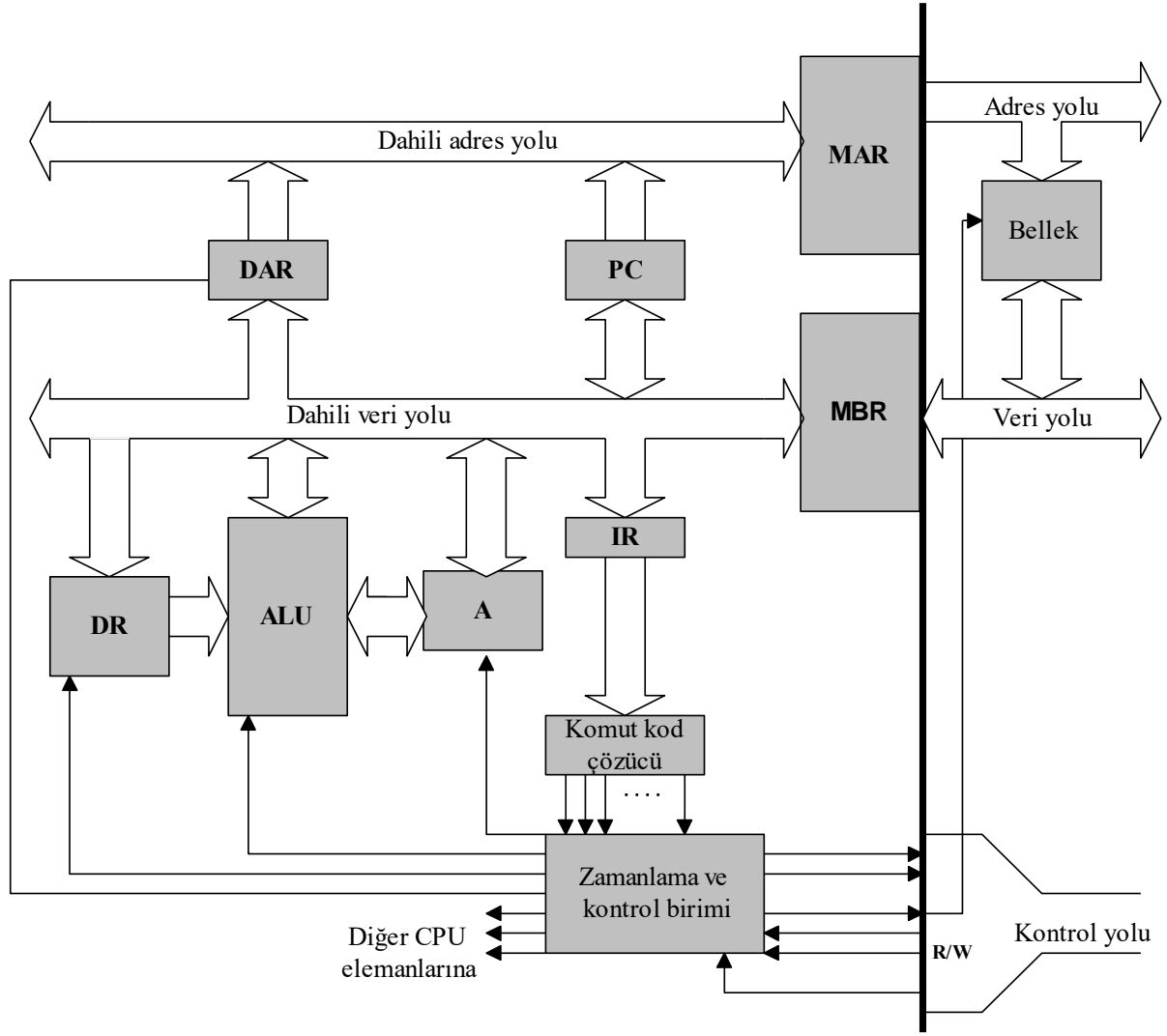
## 80X86 Ailesinin Gelişimi

### 8086 Mikroişlemcisinin Register Yapısı ve Özellikleri.

- 16 bitlik register
- 16 bitlik databus
- 20 bitlik adres bus
- $2^{20} = 1$  Megabyte hafıza adresler
- $2^{16} = 64$  Kbyte'lık giriş/çıkış portu vardır
- Sadece “Gerçek Mod” çalışma biçimi vardır

### 1.4 8-Bitlik Mikroişlemciler:

Basit bir işlemci kaydediciler, aritmetik-mantık birimi ve denetim birimi olmak üzere 3 ana bölümden meydana gelmiştir.



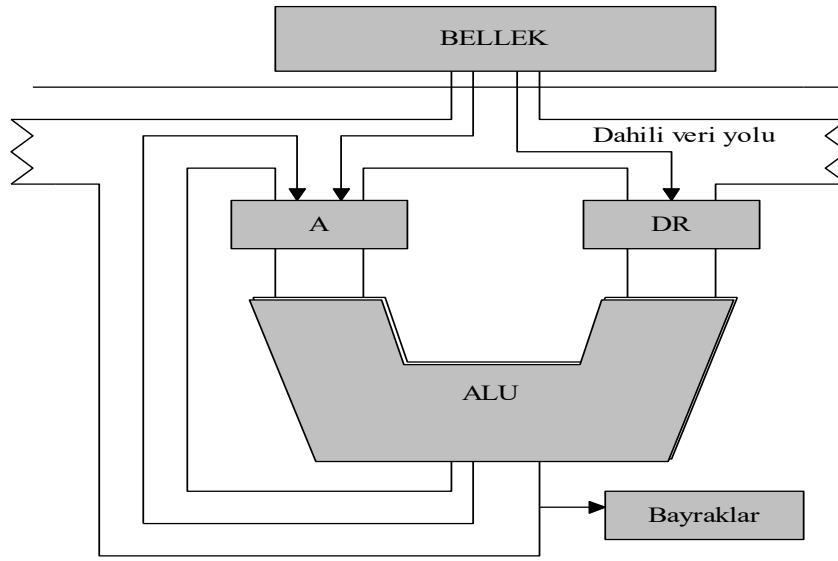
Basit bir 8-bitlik işlemcinin yapısını oluşturan ana birimler

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



kaydedicisi), DAR(Data Address Register- Veri adres kaydedicisi) ve DR (Data register- Veri kaydedicisi) olarak ele alınabilir.

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



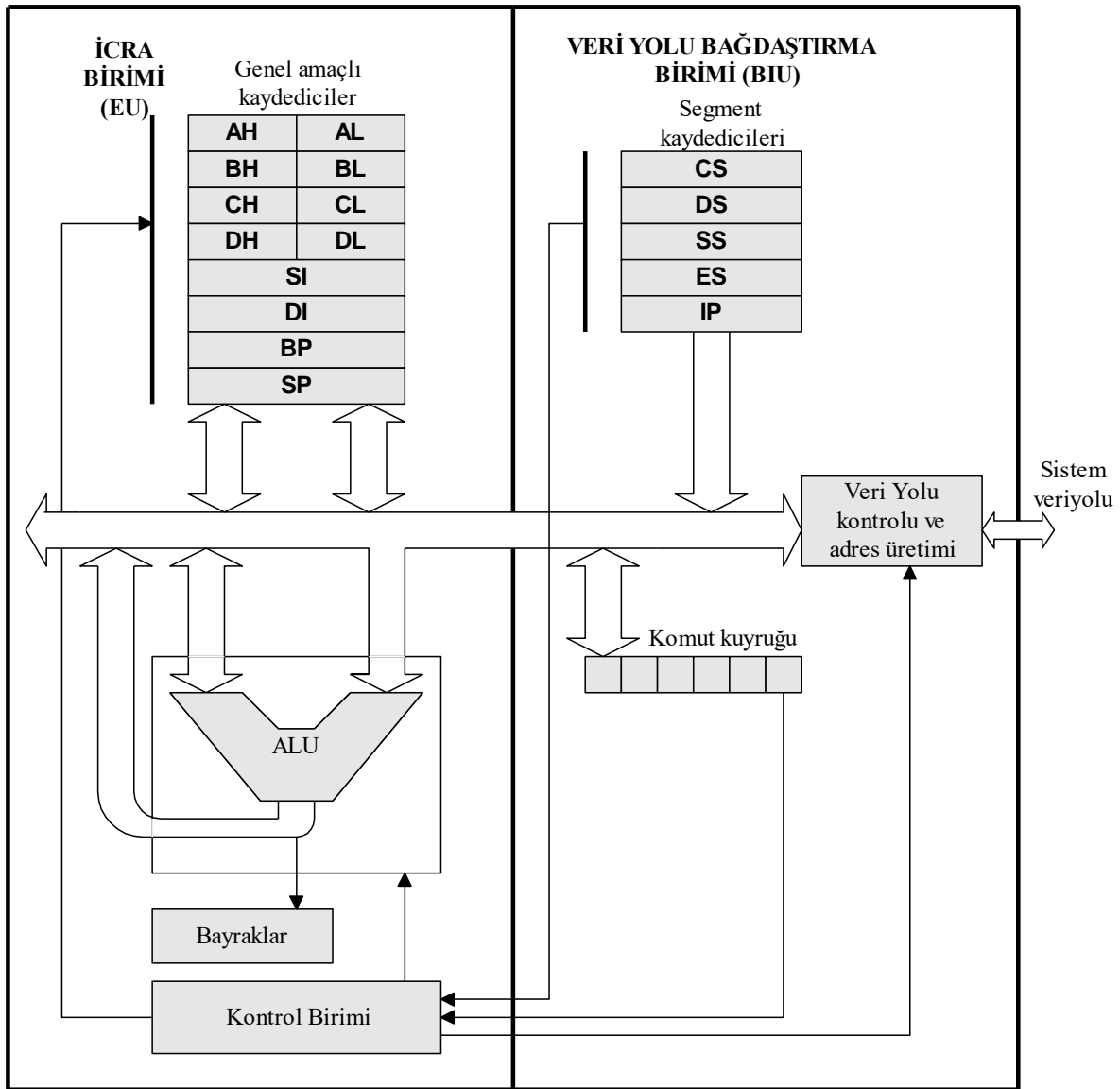
Aritmetik ve mantık birimi

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

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

## 1.5 16-Bitlik Mikroişlemciler:

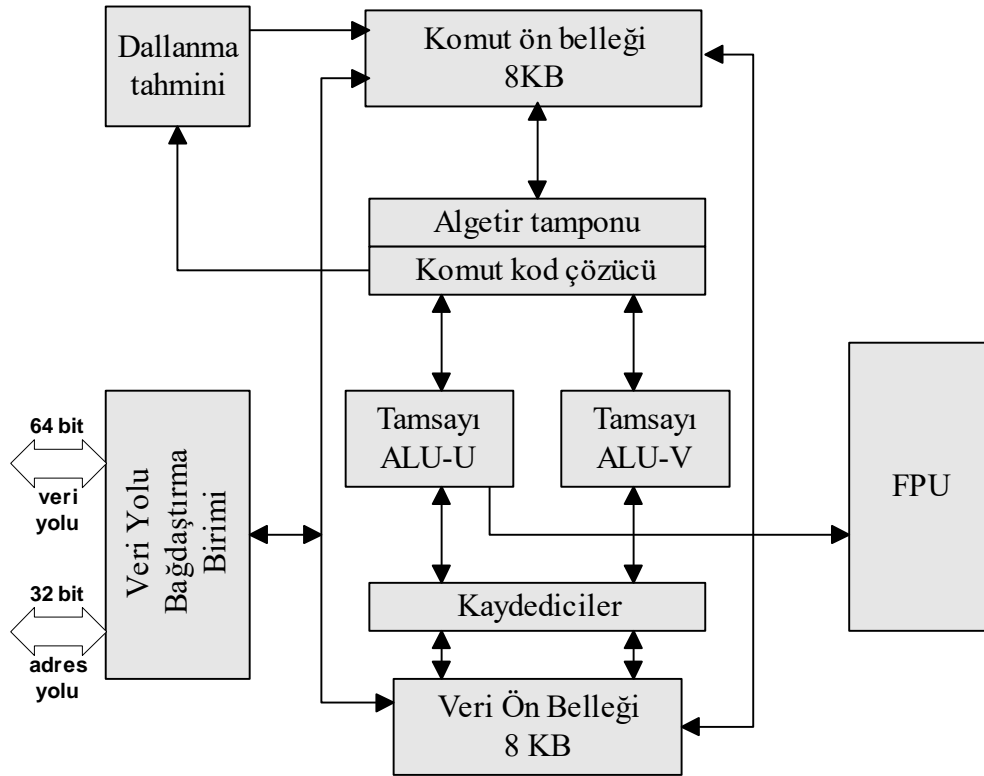
16-bitlik mikroişlemciler basit olarak 8-bitlik mikroişlemcilerde olduğu gibi , Kaydediciler, ALU ve Zamanlama-Kontrol birimine sahiptir. Fakat mimari yapısı çoklu görev ortamına uygun hale getirildiğinden, işlemci içerisindeki bölümlerde fonksiyonel açıdan 2 mantıksal bölümden oluşurlar. Bu birimler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



16 bitlik mikroişlemci mimarisi

## 1.6 32-Bitlik Mikroişlemciler:

3. kuşak mikroişlemcilerdir. Diğerlerinden farklı olarak içerisine FPU (Floating Point Unit- Kayan nokta birimi) denilen ve matematik işlemlerinden sorumlu olan bir birim eklenmiştir. Bu gelişmiş işlemci 64-bitlik geniş bir harici veri yoluna sahiptir. Geniş veri yolu, işlemcinin bir çevrimlik zamanda daha çok veri taşınması ve dolayısıyla yapacağı görevi daha kısa zamanda yapması demektir. Bu, işlemcinin bir tıklanmasıyla, işlemci ile bellek arasında veya işlemci ile G/Ç birimleri arasında, 8-bitlik bir işlemciye göre 8 kat fazla bilgi taşınması demektir.

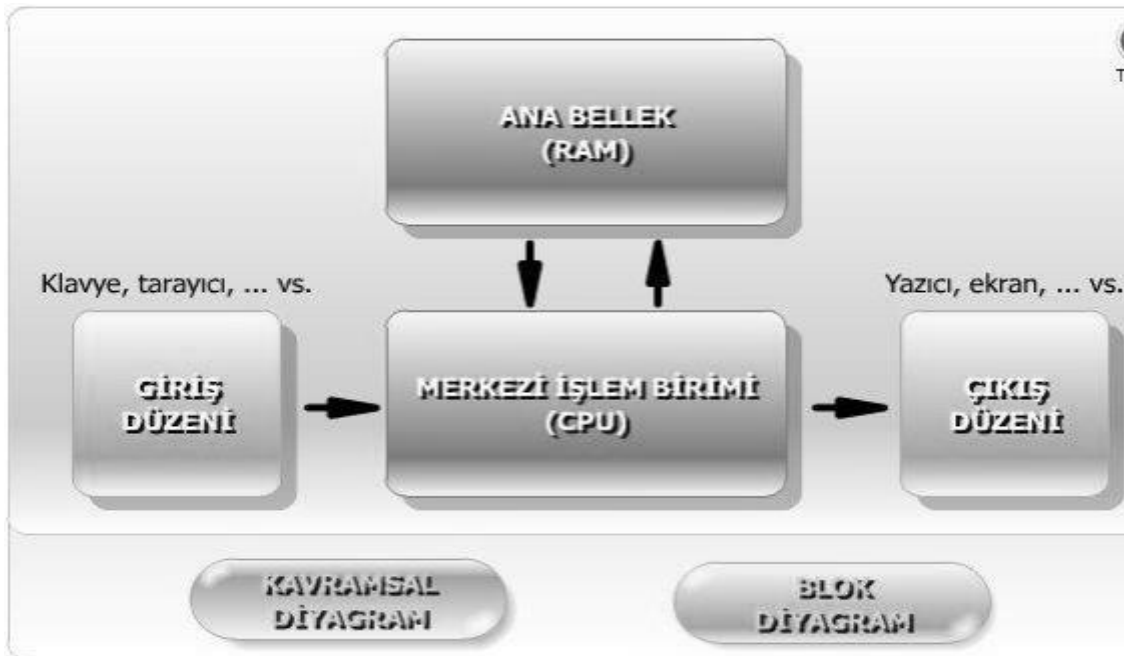
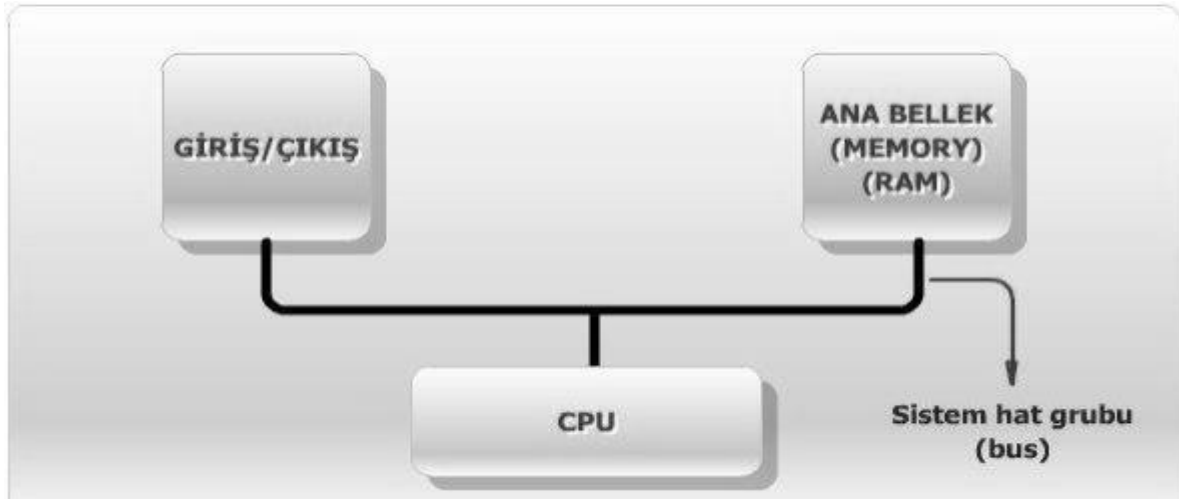


32-bitlik mikroişlemci mimarisi

## 2 BÖLÜM-2 SAYISAL LOJİK DEVRELERİN ÖZETİ

### 2.1 Sayısal Bilgisayar

Sayısal bilgisayar, basitçe, ikili (0 ve 1) sayılar ile çeşitli hesaplamaları yapan sistemdir. İkili sayı sistemi "0" ve "1"den oluşur. Aşağıda sayısal bir bilgisayarın kavramsal ve blok diyagramı gösterilmiştir (Von Neumann makinası).



Sayısal bilgisayar merkezi işlem birimi (CPU: central processing unit), bellek birimi (RAM: random access memory) ve giriş-çıkış biriminden oluşur (IOP: giriş-çıkış işlemcisi). CPU aritmetik ve lojik birimi, hızlı bellekleri (registers) ve kontrol devrelerini içerir. RAM

komutların ve verinin saklanmasıdır. IOP ise bilgisayar ile dış dünya arasında bilgi, veri alışverişini sağlar.

Bilgisayar, program yoluyla verileri (data) işler. Yazılan program (tanımlanan algoritma), çözülmesi istenen probleme uygun bir çözüm üretir. Diğer adı veri işlemedir (data- processing).

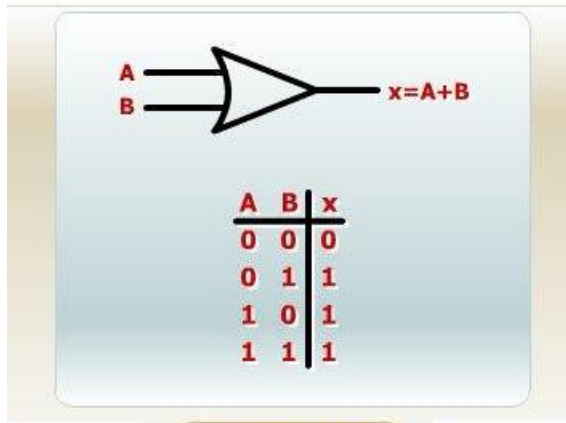
Sayısal bilgisayarı veya sadece bilgisayarı, yazılım ve donanım olmak üzere iki temel kısımdan oluşmuş kabul edebiliriz. Donanım fiziksel gerçekleştirilebilen elemanlardır: lojik kapılar, FF'ler, sayısal komponentler (tasarlanmış sayısal sistem elemanları, örneğin dekode, enkode, multiplekser-çoğullayıcı). Yazılım ise programlardan oluşur.



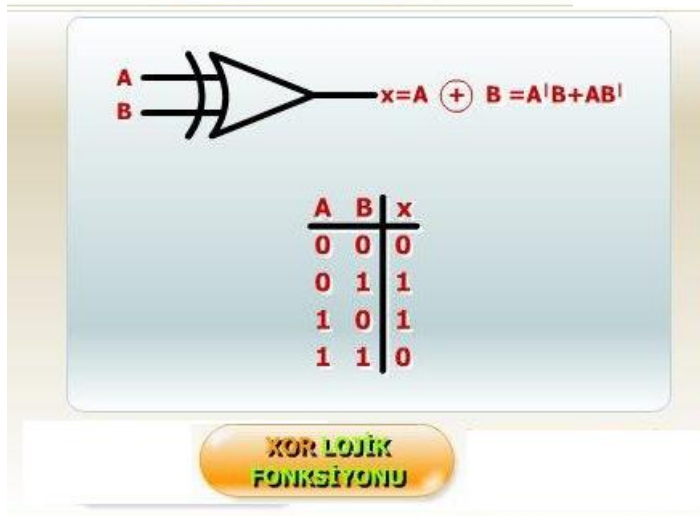
Bilgisayar organizasyonu donanım komponentlerinin interaktif olarak nasıl çalıştığını tanımlar. Bilgisayar mimarisi, yapının kullanıcı tarafından görülen kısmı ile ilgilidir, örneğin komut (instruction) formatları, adres modları (addressing modes).

## 2.2 Lojik Kapılar

Lojik kapılar sayısal tasarımın temel elemanlarıdır. Tipik örnekleri arasında AND, OR, NOT, buffer, XOR' u sayabiliriz. İkili (0,1) bilgi girişlerinden yeni bir ikili (0,1) çıkış bilgisi üretirler. OR ve XOR örnekleri aşağıda gösterilmiştir.



OR LOJİK FONKSİYONU



XOR LOJİK FONKSİYONU

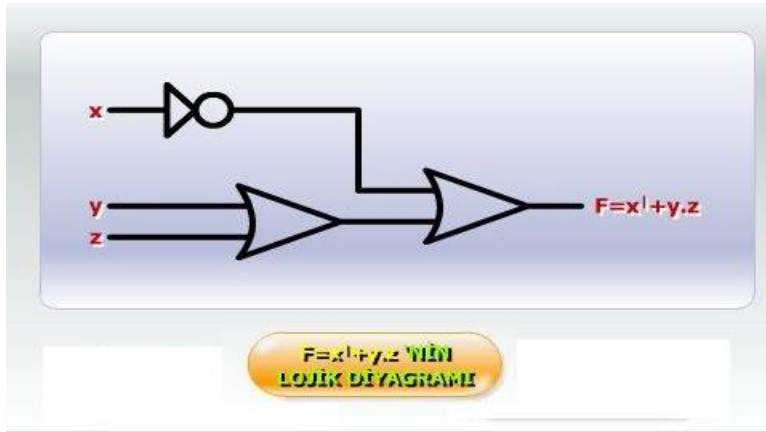
Lojik kapıların giriş-çıkış ilişkisi, “doğruluk tablosu” (truth table) aracılığıyla gösterilir. Doğruluk tablosu ikili giriş değişkenleriyle çıkış değişkeni arasında dönüşümü sağlayan tablodur.

### 2.2.1 Boole Cebri

“Boole Cebri” sayısal devrelerin analiz ve tasarımını sağlayan matematiksel teoridir. Sayısal bilgisayar devreleri uygulamasında, ikili değişkenler üzerinde tanımlanan lojik operasyonları gösterir. Örneğin,

$$F=x^1+y.z$$

Üç değişkenli (x, y, z) bir F Boole fonksiyonunu tanımlar. Doğruluk tablosu yoluyla F Boole fonksiyonunun x, y, z değişkenlerinin çeşitli değerlerine göre alacağı değerler tanımlanır ve lojik diyagram ile lojik kapılar cinsinden şekillendirilebilir.



x	y	z	F
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

**F = x' + y.z TİN  
DOĞRULUK TABLOSU**

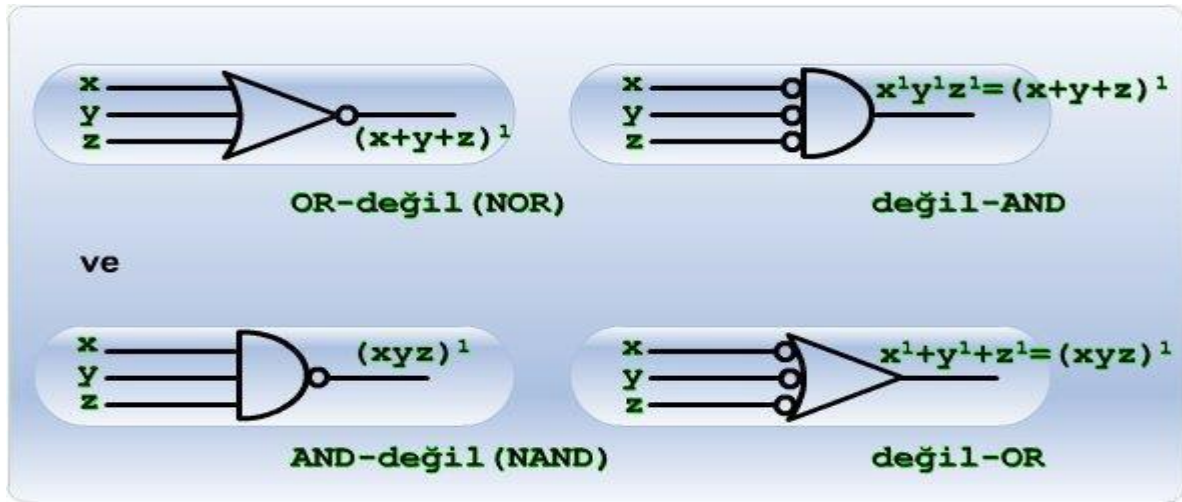
Boole cebirinin temel eşitlikleri (örn:  $0+1=1$ ,  $x.1=1$ , ...) ve De Morgan teoremi devrelerin gerçekleştirilmesinde kullanılan temel kurallardır.

### 2.2.2 DeMorgan Kuralı

DeMorgan kuralı NOR ve NAND kapıları ile aşağıdaki dönüşümleri sağlar.

$\text{NOR} \Rightarrow (x + y)' = x' y'$ $\text{NAND} \Rightarrow (x y)' = x' + y'$
--

Bu durumda; aşağıdaki bağlantılar elde edilir.

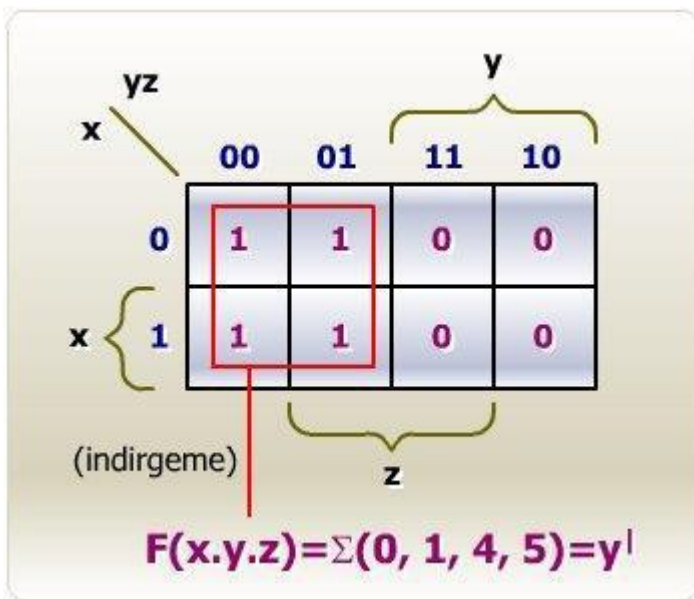


### 2.2.3 Karnaugh Haritasının Basitleştirilmesi

Boole fonksiyonu küçükterimlerin (minterm) kombinasyonu cinsinden yazılıp basitleştirilebilir. Bu işlem tasarlanan lojik devrelerin minimum elemanlı tasarımı için kullanılır. “n” değişkenli Boole fonksiyonu  $2^n$  küçükterim içerir. Örneğin  $n=3$  değişkenli K-map 8 küçükterim ile gösterilir.

$$F(x, y, z) = \Sigma (0, 1, 4, 5)$$

Komşu kareler Boole fonksiyonunun sadeleştirilmesinde kullanılır. Böylece en az sayıda eleman içeren bir devre tasarlanır.

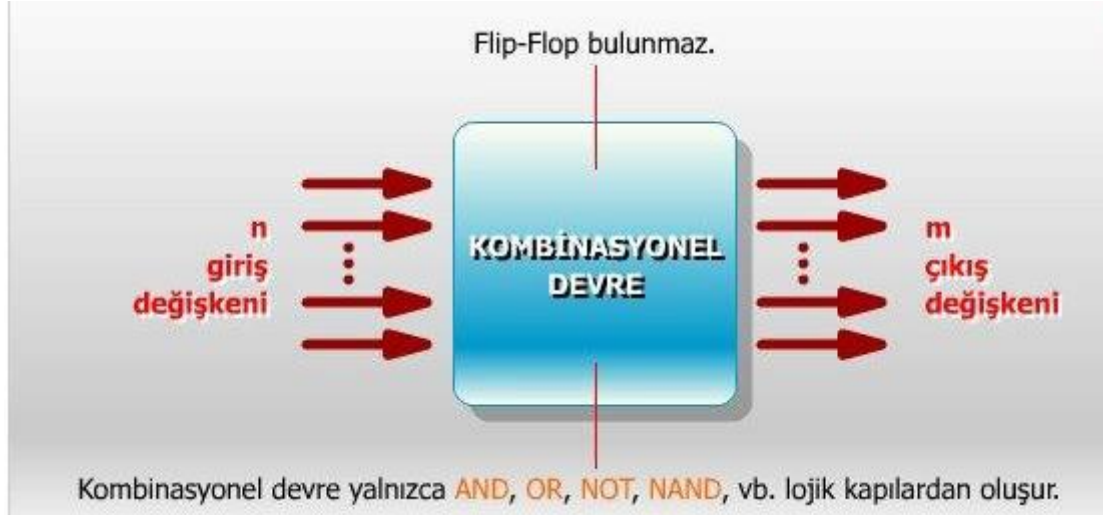


$F(x,y,z) = \Sigma (0,1,4,5)$  K-map gösterilimi



## 2.3 Kombinasyonel Devreler

Kombinasyonel devre lojik kapıların, giriş ve çıkışların bağlanmasından oluşan devredir. Geribesleme ve Flip-Flop elemanı bulundurmaz. Çıkışlar sadece giriş elemanlarına bağlıdır.



Kombinasyonel devrenin analiz ve tasarımı doğruluk tablosu yoluyla yapılabilir. Analiz evresinde, her bir devre çıkışı, giriş değişkenleri cinsinden yazılır. Tasarım evresinde, verilen problem giriş ve çıkış değişkenleri cinsinden tanımlandıktan sonra, doğruluk tablosu ile giriş-çıkış bağıntısı belirlenir. Sonra, K-map yoluyla basitleştirme (minimum elemanla devre gerçekleştirme) yapıp, devre gerçekleştirilir.

### 2.3.1 Yarı-Toplayıcı (HA:Half Adder)

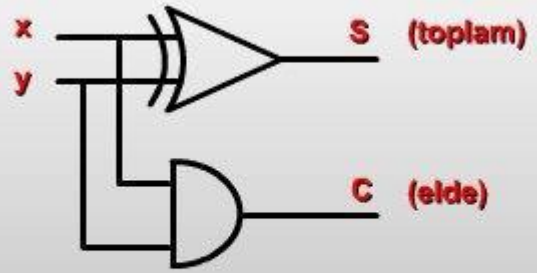
Tam-toplayıcı (FA:full adder) için ise S (toplam) ve C (elde) çıkışları x,y,z girişleri cinsinden aşağıdaki lojik eşitliklerle gösterilir.

$$S = x \oplus y + z$$

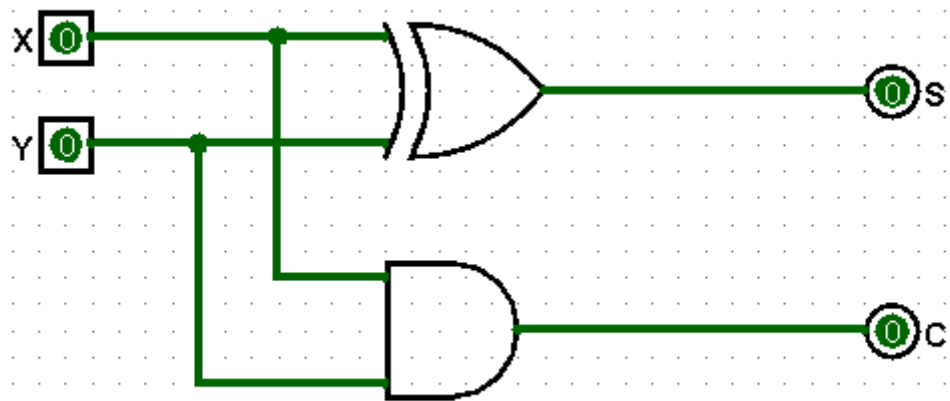
$$C = xy + (x \oplus y)z = xy + (x' + (x'y + xy'))z$$

x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

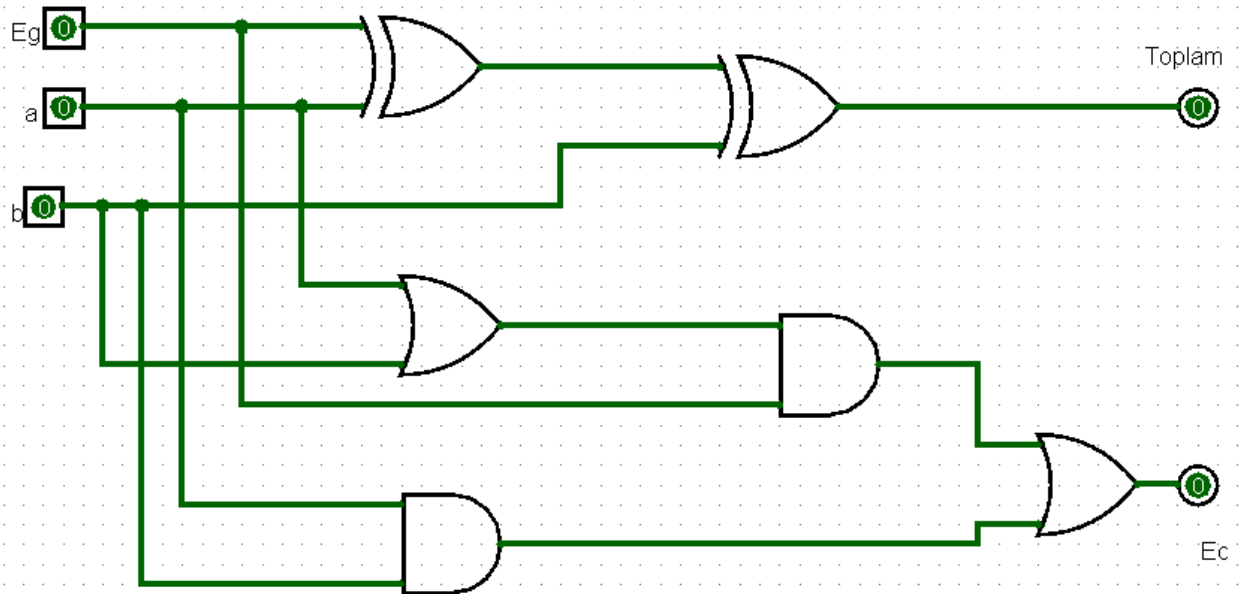
Doğruluk tablosu

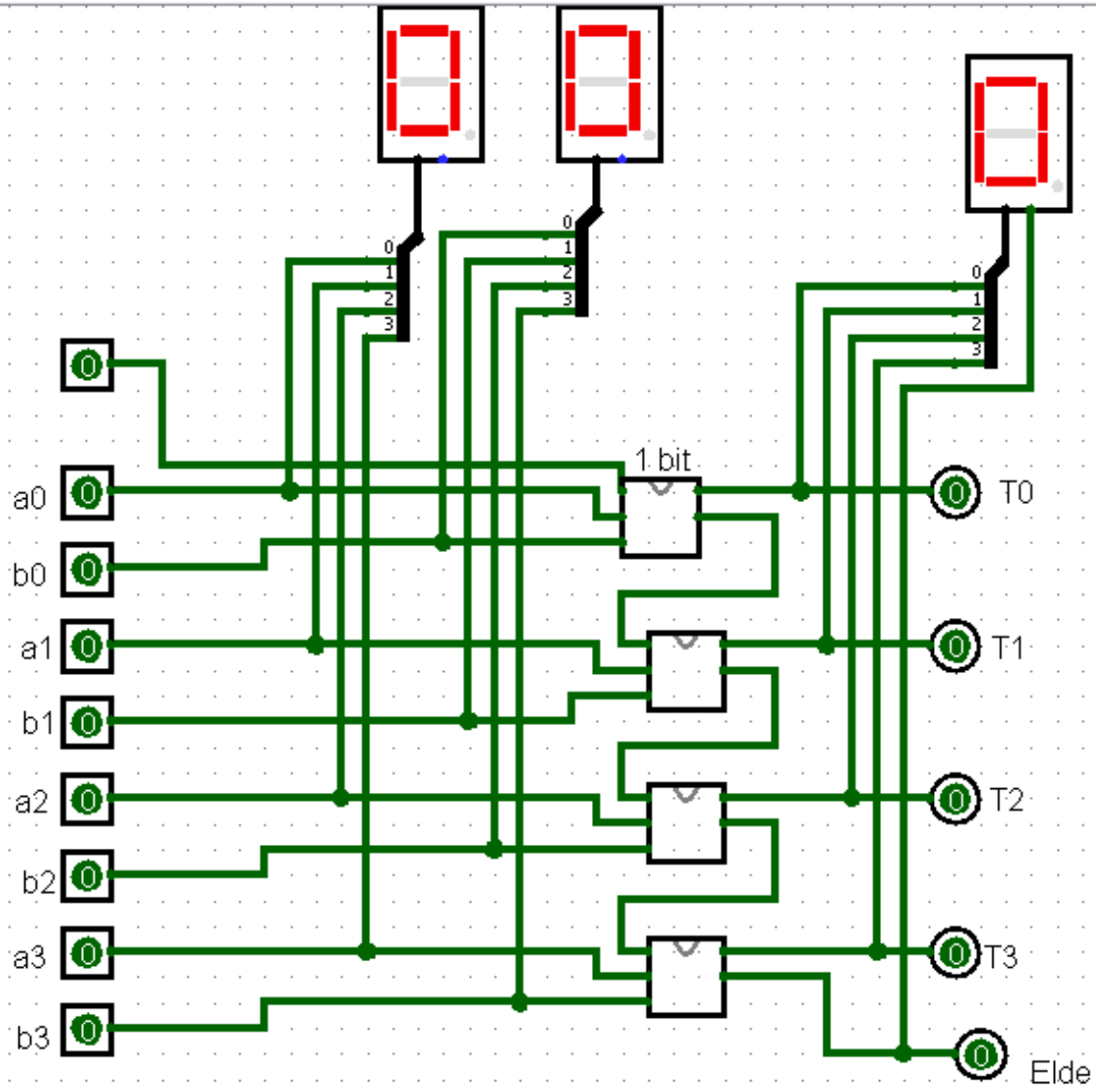


Lojik diyagram



### 1 Bit Tam Toplayıcı



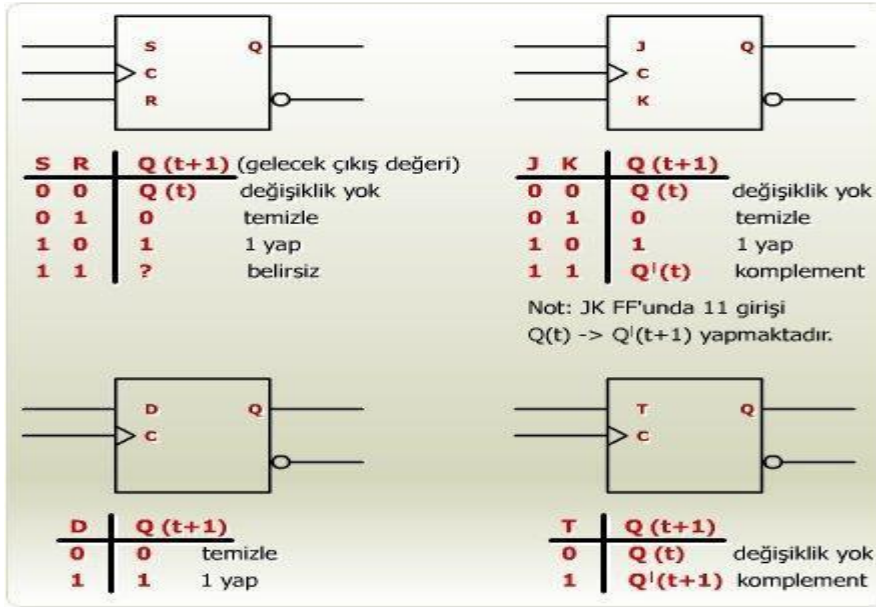


## 4 Bitlik Toplayıcı

### 2.4 Flip-Floplar (FF)

Bu devrelerde elemanlar hafıza (memory) içerirler. Ardışıl devre olarak adlandırılırlar. Saat darbeleri (clock pulses) ile çalışırlar. Eşzamanlı (senkron) ve eşzamanlı olmayan (asenkron) olmak üzere iki tipleri vardır. Eşzamanlı olanlar saat darbeleri ile çalışırlar, eşzamanlı olmayanlar ise giriş büyüklüğünün devreye ulaşması ile yeni çıkış değerlerini alırlar.

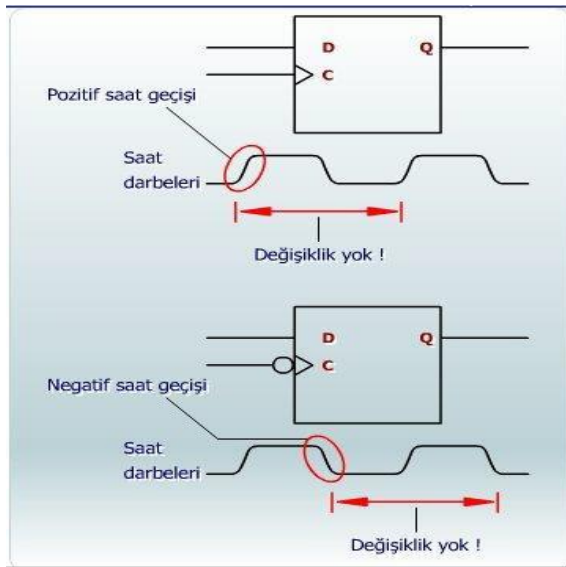
Dört tip FF elemanı vardır: SR, JK, D, T. FF elemanı giriş veya girişlerine uygulanan ikili değer ile şimdiki durumdan (present state) gelecek duruma (next state) geçer. Aşağıda FF'lerin uyarıma tabloları verilmiştir.



### FF'lerin uyarma tabloları

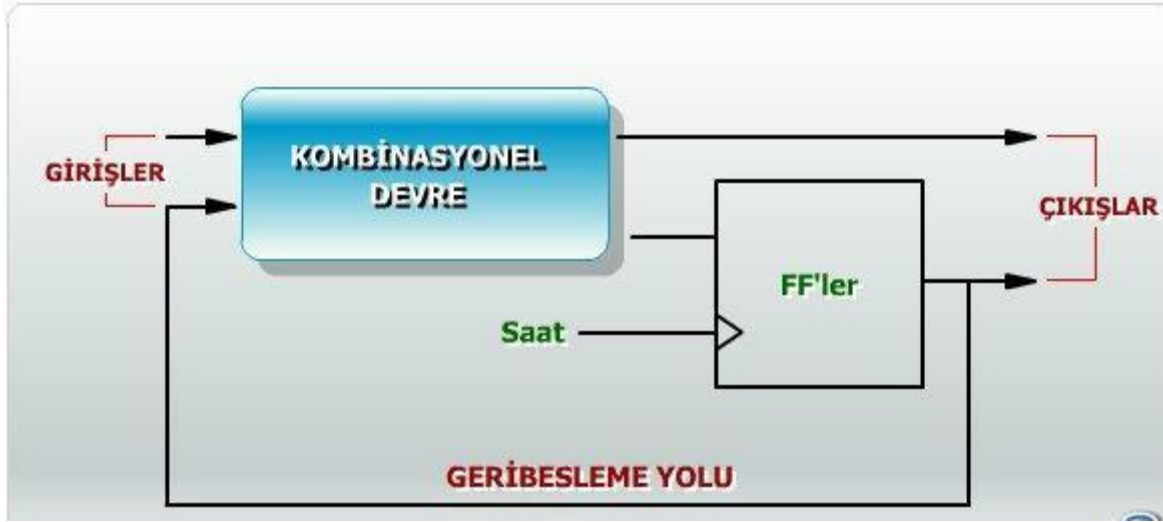
#### 2.4.1 Kenar Tetiklemeli ve Efendi-Köle (Master-Slave) Tipi FF'ler

Kenar tetiklemeli ve Efendi-köle (Master-slave) tipi FF'ler yaygın olarak kullanılan tiplerdir. Kenar tetiklemeli FF, yalnızca saat darbelerinin uygulanması ile durum değiştirir. Saat darbelerinin geçişi sırasında durum değişir; diğer anlarda durum sabit kalır. Efendi- köle (master-slave) tipi FF' de giriş değeri ilk saat darbesi geçişinde master çıkışında olur. İkinci darbe geçişinde aynı çıkış köle FF çıkışı olur. Aşağıda kenar tetiklemeli FF'nin çalışması açıklanmaktadır.



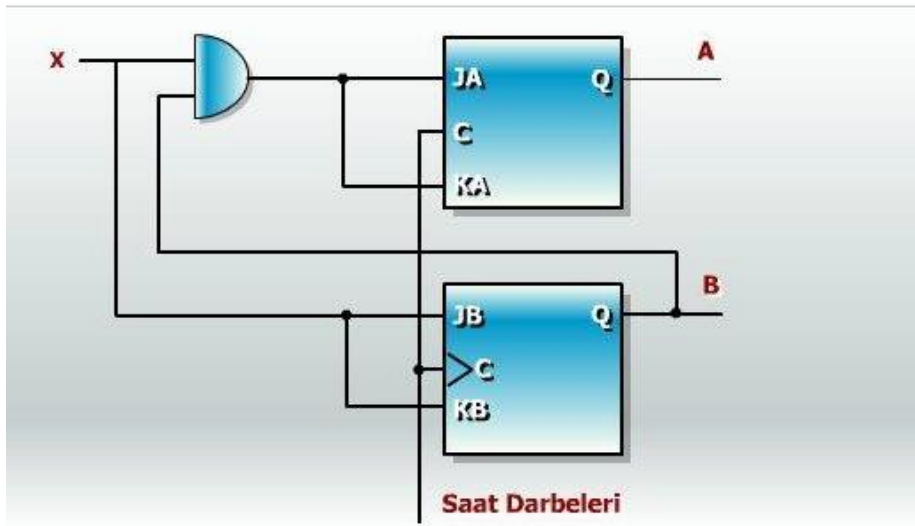
## 2.5 Ardışıl Devreler

Ardışıl devreler FF'ler ve lojik kapıların bağlanmasından oluşur. Aşağıda ardışıl devrenin blok diyagramı gösterilmiştir. Şekilde FF' nin çıkışından alınan geri beslemenin sisteme giriş olarak uygulandığı izlenebilir.



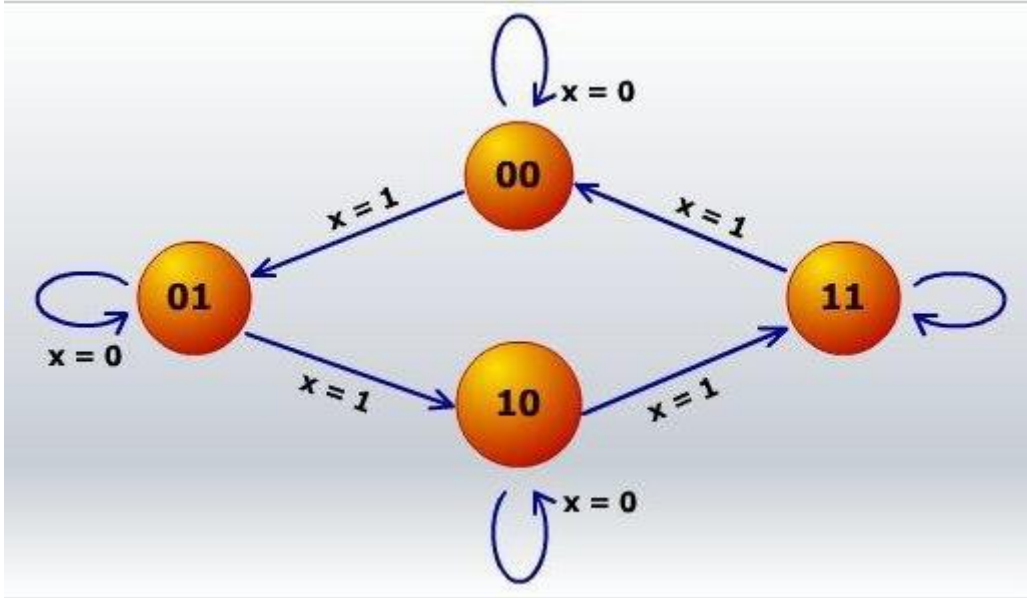
Ardışıl devrenin analizi işlemi, verilen lojik diyagramdan yola çıkarak, giriş eşitliklerinden, uyarma tablosu yoluyla durum diyagramının elde edilmesi olarak özetlenebilir. Tasarım sırasında verilen durum diyagramından lojik diyagramın elde edilmesidir. Tasarımın ara işlemi olarak uyarma tablosu kullanılır. İstenen giriş-çıkış ikili bağıntısını sağlayan FF' lerin giriş eşitlikleri elde edilir.

Örnek olarak verilen bir devrenin analizini yapalım.



$$\begin{aligned} J_A &= Bx \quad K_A = Bx \\ J_B &= X \quad K_B = X \end{aligned}$$

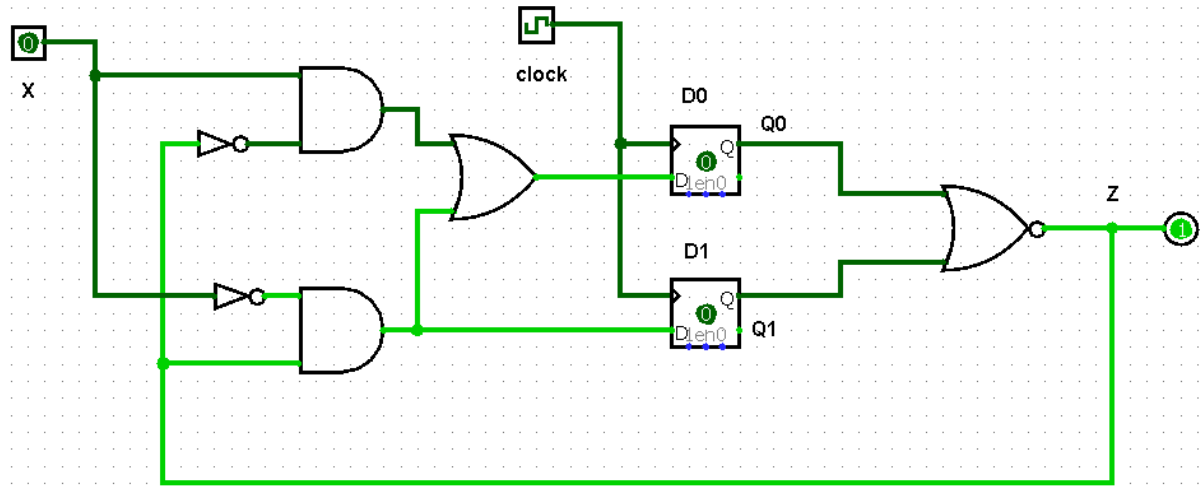
Sonra x giriş değerleri için durum diyagramı çizelim (durum tablosu da oluşturulabilir). Dört durum arasında geçişler x değerine göre olacaktır.



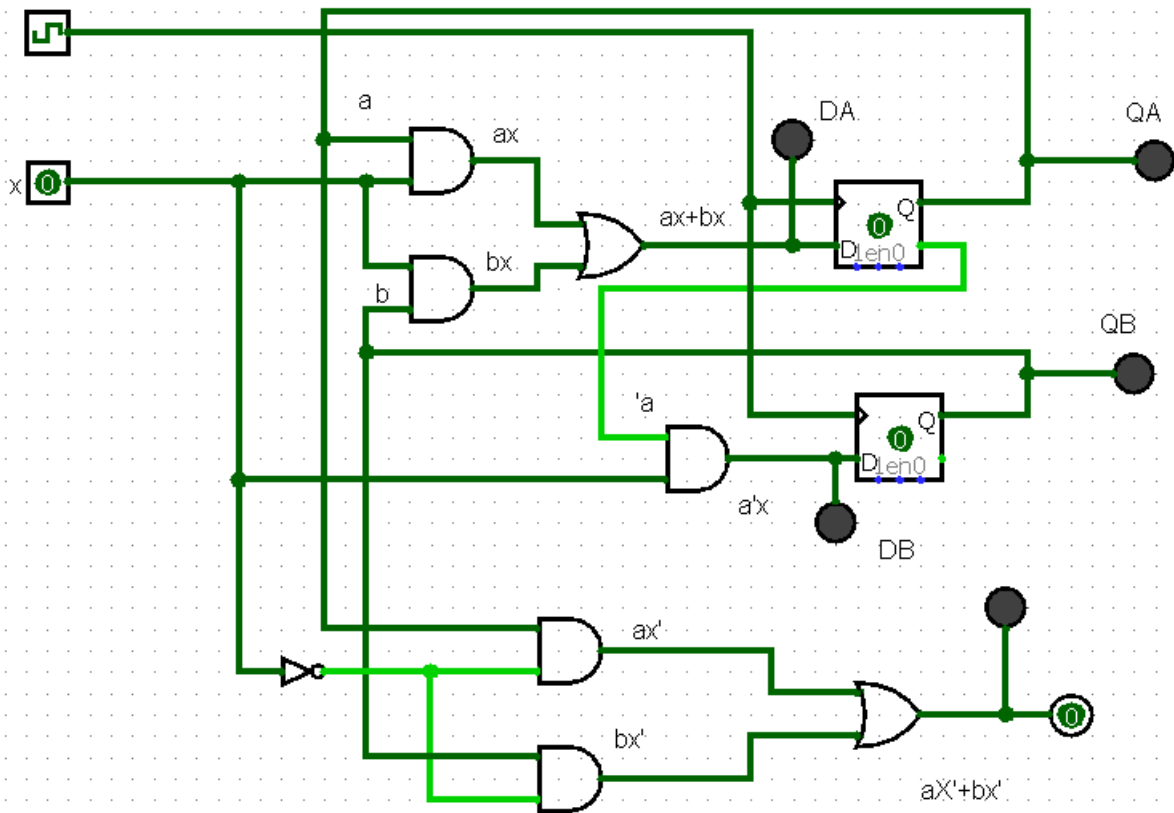
**Sonuç:** Yukarda Lojik diyagramı gösterilen devre bir 2-bit ikili sayıcı devresidir.

Analiz örneğinde verilen bir lojik devrenin FF'lerinin giriş eşitliklerinin bulunması ve çıkış eşitliğinin elde edilmesi hatırlatılmıştır. Tasarım örneğinde ise 2-bit eşzamanlı sayıcı gösterilmiştir.

### Ardışıl Devre Örneği



### Ardışıl Devre Örneği 2





## 3 BÖLÜM SAYISAL BİLEŞENLER

### 3.1 Tümdevrelerin Tanıtımı

Tümdevreler (integrated circuits, chip) çok sayıda lojik elemanın gerçekleştirildiği, küçük, yarı iletken sayısal bileşenlerdir. Küçük bir paket içinde yüzlerce, binlerce lojik kapı, FF gibi elemanı gerçekleştirip, bağlantıları bacaklar ile dışarıya aktararak elde edilir.



Tümdevreler kapasitelerine göre yukarıdaki gibi çeşitlendirilebilir. KÖB' de 10'dan az sayıda eleman bir paketin içinde bulunur. OÖB' de bu sayı 10 ile 200 arasındadır. GÖB' de ise eleman sayısı 200 ile birkaç binler arasında değişir. Son olarak, ÇGÖB' de ise bir paketteki eleman sayıları kolaylıkla yüzbinler mertebesinde bir fiziksel örnek vermek gerekirse, tırnak yüzeyi boyutundaki yarı iletken üzerinde 400-500 binler mertebesinde lojik kapı gerçekleştirilmektedir.

Tümdevreler birkaç tipik teknikle üretilmektedir: bunlar TTL (transistör, transistör logic- transistör, transistör lojik), ECL (emitter coupled logic-emetör kuplajlı lojik), MOS (metal oxide semiconductor-metal oksit yarı iletken), CMOS (complementary metal oxide semiconductor-komplementer MOS). TTL teknolojisi yıllardır standart olan, 5 volt besleme gerilimi ve lojik 0 ve 1 seviyeleri için 0 volt ve 3.5 volt kullanan, NAND kapıları temel olarak elde edilmiş bir teknolojidir. ECL ise hızın önemli olduğu uygulamalar, örneğin süper bilgisayar ve işaret işleme donanımlarında kullanılan, transistörleri doyuma (saturasyona) girmeyen bir teknolojidir. MOS teknolojisi bileşen yoğunluğunun fazla olduğu bir teknolojidir. n-kanallı (NMOS) ve p-kanallı (PMOS) olmak üzere yalnızca bir çeşit akım taşıyıcının kullanıldığı çeşitleri vardır. CMOS teknolojisi ise bugün yaygın olarak kullanılan, üretimi basit ve düşük güç tüketimi olan bir teknolojidir.

Yukarda belirtilen teknolojilerle üretilen sayısal bileşenler bilgisayar tasarımında çeşitli fonksiyonların sağlanmasında kullanılmaktadır. Aşağıda ise kullanılan bileşenleri daha detaylı olarak inceleyeceğiz.

### 3.2 Dekoder-Enkoder

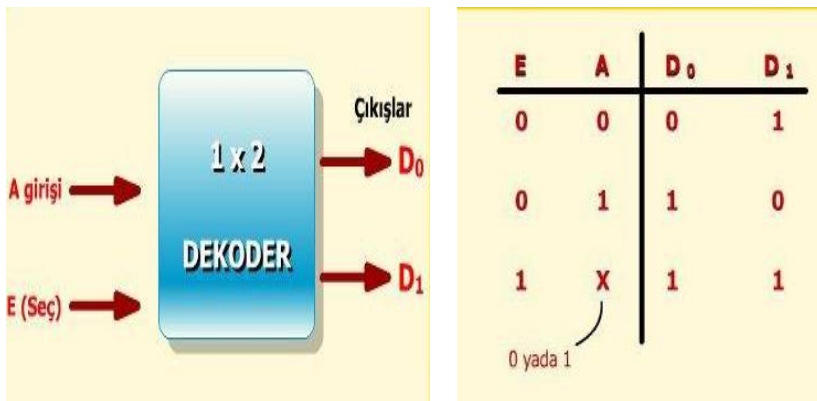
Dekoder ve enkoder, bilgisayarda  $(n-m)$  bit dönüşümünü sağlayan Kombinyonel devrelerdir. Bilindiği gibi, bilgisayar ikili sayılar üzerinde hesap yapmaktadır. Giriş ve çıkışta farklı sayıda bit elde etmek önemli bir işlemdir. Bu ise  $n$  bit kodlanmış bilginin  $m=2^n$  bir çıkış bilgisi veya tam tersi bir  $2^n$  bit girişin bir  $n$  bit çıkış bilgisi haline çevrilmesi işlemidir. Burada giriş çıkış kodlamasında

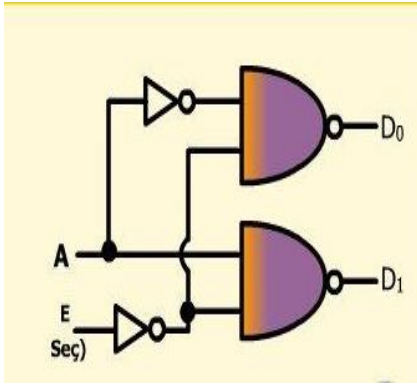
2'nin kuvvetlerinin kullanıldığı gözlenmelidir. Ayrıca bu devreler, tümdevre gerçekleştirilmesi ile kullanıldığı için Enable-E girişi ile istenilen devrenin seçimi sağlanır:  $E=0$  tümdevrenin seçilmemesine,  $E=1$  ise seçilip, işleminin kullanılmasına neden olur.

#### 3.2.1 Dekoder

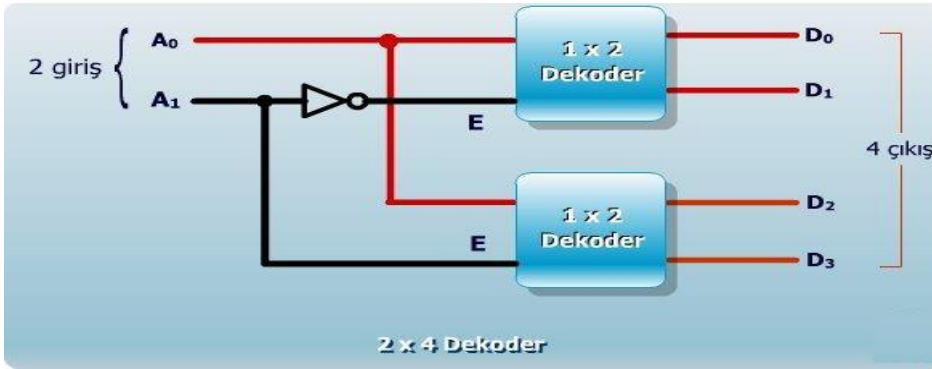
Dekoder  $n$  girişli ve  $m=2^n$  çıkışlı bir elemandır.  $n \times m$  dekode olarak da adlandırılır. Amacı,  $n$  girişte verilen ikili bilgiye göre  $2^n$  çıkıştan birini üretmektir. Aşağıda  $1 \times 2$  dekode blok diyagramı, doğruluk tablosu ve NAND kapılı gerçekleştirilmesi (lojik diyagramı) gösterilmiştir. Dekoderin AND, OR, NOR kapıları ile de gerçekleştirilebileceği unutulmamalıdır.

$1 \times 2$  dekode blok diyagramı, doğruluk tablosu ve NAND kapılı gerçekleştirilmesi girişte uygulanan değere göre (0 veya 1) çıkışlardan birisi seçilir: 0 uygulanırsa 0 çıkışı, 1 uygulanırsa 1 çıkışı seçilir.





Dekoderin aynı zamanda 2x4, 3x8, 4x16 ve daha çok sayıda bit dönüşümü yapabileceği göz önüne alınmalıdır. Bir uygulama olarak 2 adet 1x2 dekoder kullanılarak 2x4 dekoderin fonksiyonunun sağlandığı (kapasite arttırımı) aşağıda gösterilmiştir.

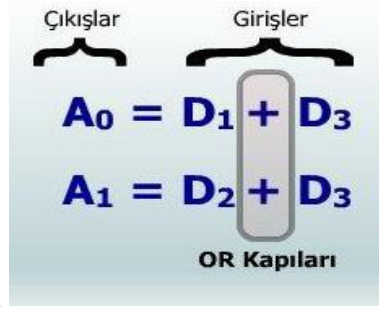


2 adet 1x2 dekoder ile 2x4 dekoder elde edilmesi

2x4 dekoderin çalışması 1x2 dekodere benzer. Bu durumda giriş sayısı 2 çıkış sayısı ise 4'tür. Girişe uygulanan ikili değere karşı düşen çıkış seçilir. Örneğin, giriş değeri 00 ise 00 çıkışı, 10 ise 10 çıkışı seçilir.

### 3.2.2 Enkoder

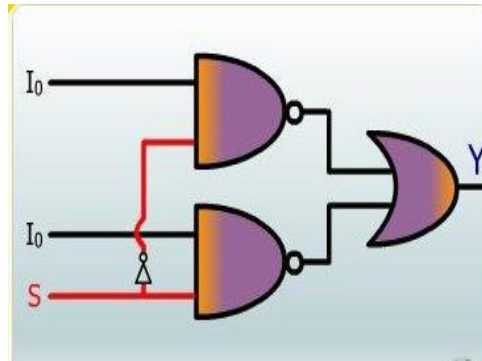
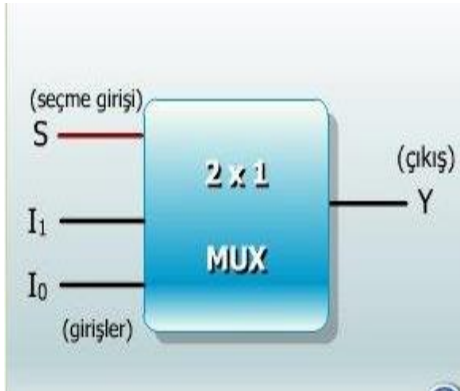
Enkoder ise  $2^n$  girişi  $n$  çıkışa dönüştüren veya dekoderin ters işlemini yapan devredir. Aşağıda 4x2 enkoderin blok diyagramı, doğruluk tablosu ve OR kapıları ile gerçekleştirilmesi gösterilmiştir. Devrede bir zamanda, yalnızca bir giriş "1" olabilir.



D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>	A <sub>1</sub>	A <sub>0</sub>
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

### 3.3 Çoğullayıcı (Mux)

Çoğullayıcı (MUX)  $2^n$  girişli ve 1 çıkışlı kombinyasyonel bir devredir. Girişlerden birinin çıkışta seçimi için kullanılır. Bu nedenle enable-E girişi yanında seçim hatları (selection lines) kullanılması gerekmektedir. 2 girişin seçimi için 1 seçim hattı, 4 girişin seçimi için ise 2 seçim hattı gerekeceği hatırlanmalıdır. Aşağıda bir 2x1 çoğullayıcının blok diyagramı, doğruluk tablosu ve gerçekleştirilmesi gösterilmiştir.



2x1 MUX'un seçim hattı hangi girişin çıkışa aktarılacağını tayin eder: Seçim hattı 0 ise 0 girişi çıkışa gönderilir, 1 ise 1 girişi çıkışa gönderilir.

S	Y
0	$I_0$
1	$I_1$

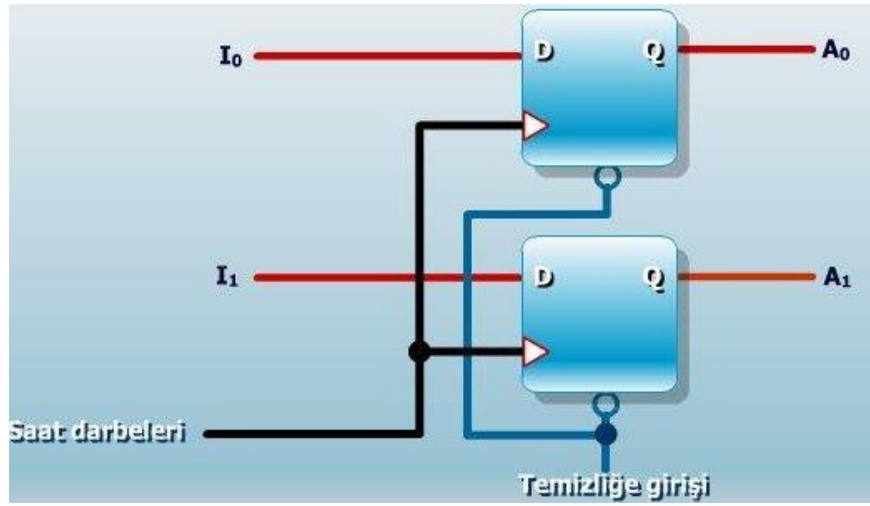
**! Uyarı:** Burada kullanılan 2 "tek" giriş hattı yerine 4'lü veya 8'li giriş hatları aynı sayıda seçim hattı ile kullanılabilir.

### 3.4 Bellek Elemanı (Register) Saklayıcı

Bellek elemanı ikili bilgi depolama özelliğine sahip bir dizi Flip-Floptan oluşur. Genellikle, 2<sup>n</sup> bit FF dizisi bu görevi görür, ikili bilgi kapıların kontrolü yoluyla FF' lere yazılır veya okunur. Bellek elemanı olarak kullanılabilen çeşitli FF' lerin tipleri arasında, D FF basitliği nedeniyle en sıklıkla kullanılanlardan birisi olmaktadır. Örneğin, 1011 4-bitlik bir bilgi olup, bunu 4 D FF' sine depo etmek istersek FF girişlerine 1011 uygulamak gerekir.

Bellek elemanlarının çıkış değerlerinin elde edilmesi okuma (read) işlemi olarak adlandırılır. Bunun için elemanın adresinin bilinmesi ve çıkışının izlenmesi gerekir. Bellek elemanına yeni bir ikili bilgi yazılması ise yazma (write) veya yükleme (load) olarak adlandırılır. Aşağıdaki şekil 2-bitlik bir  $I_0 I_1$  ikili bilgisinin 2 D FF' sine doğrudan ve bir yükle girişi yoluyla kontrollü bir şekilde yüklemesini göstermektedir.

Saat darbelerinin (clock- C) düzenli olarak FF' lere uygulandığı gözlenmelidir. Yükle girişi 1 değerini aldığı anda, giriş bilgisi FF' lere yüklenir.



### D FF'e I0 I1 ikili bilgisinin yüklenmesi

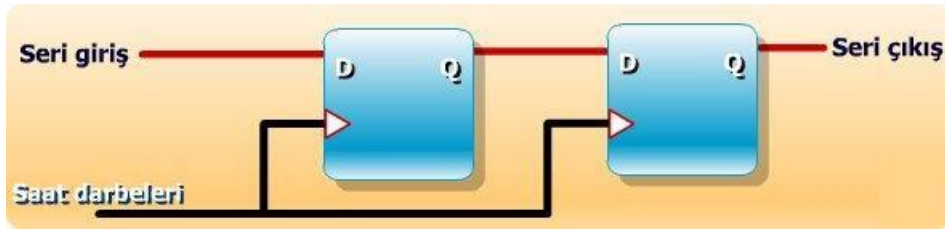
Bellek elemanın içeriğini kaybetmemesi için Yükle giriři aktive edilmediđi zamanlarda, Saat darbeleri çıkıřa ve giriře uygulanır. Her saat darbesinde bellek içeriđi "tazelenmiř" (refresh) olur.

### 4 Bitlik Saklayıcı Tutucu Register

The diagram illustrates a 4-bit shift register circuit. It consists of four D flip-flops, each with a clock input (clk Clr(t)+T) and a data input (D, eh0). The outputs of the flip-flops are labeled Q0, Q1, Q2, and Q3. The initial state of the register is 0001, as shown by the values in the flip-flop outputs. The circuit is controlled by a 'Load' signal and a clock signal.

### 3.5 Kaydırmalı Bellek Elemanı

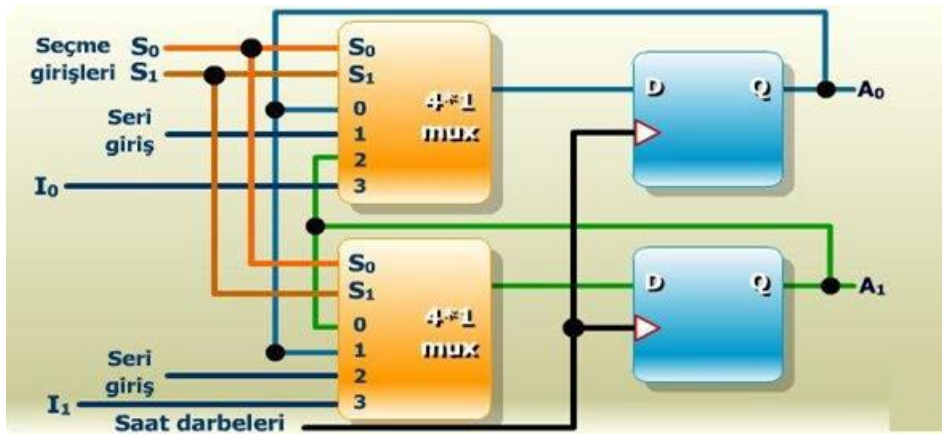
İkili bilgiyi iki yönde kaydırma özelliğine sahip bellek elemanına kaydırmalı bellek adı verilir. Bütün bellek elemanlarına ortak saat darbeleri uygulanır ve kaydırma yönüne göre ikili bilgi bir FF ileri veya geriye kayar. Aşağıdaki şekilde 2-bit kaydırmalı bellek elemanı kavramı açıklanmaktadır.



2-bit kaydırmalı bellek elemanı kavramı

### 3.6 Bit İki Yönlü Kaydırmalı Paralel Yüklemeli Bellek Elemanı

Aşağıda 2-bit iki yönlü kaydırmalı, paralel yüklemeli bellek elemanı ve çalışma prensibi anlatılmıştır. Girişteki MUX elemanının, 4 girişindeki farklı bilgileri seçim hatları ile seçilerek dört fonksiyonu gerçekleştirilmiştir: bilgi tazeleme (değişiklik yok), sağ kaydırma, sol kaydırma ve paralel yükleme.  $S_1S_0$  seçim hatları devrenin fonksiyonunu belirler.



2-bit İki yönlü kaydırmalı, paralel yüklemeli bellek elemanı

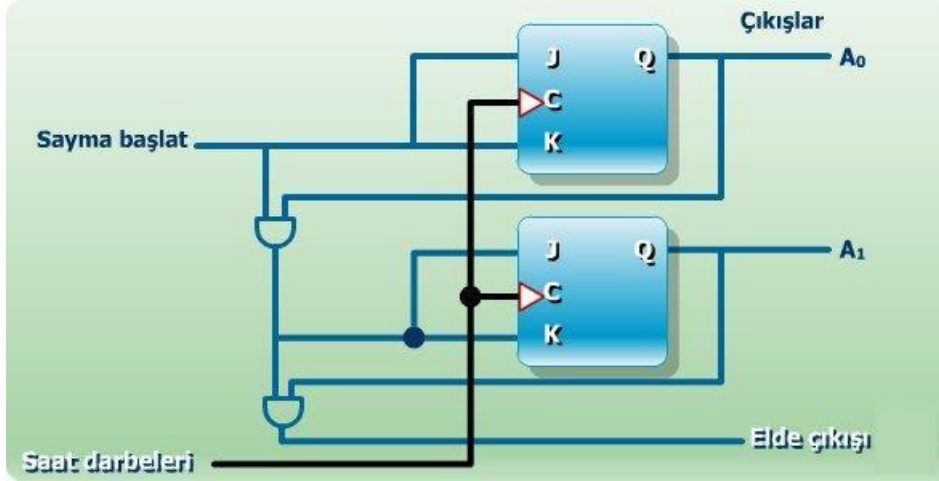
Şekil dikkatle incelenirse sağa ve sola kaydırma işlemi sırasında iki yanda bulunan FF'lerin çıkışlarının seçilen MUX girişine uygulandığı görülebilir. Örneğin,  $A_2$  bitinin sağa kaydırılması sırasında  $A_2$ 'ye ait 4x1 MUX girişine  $A_1$  çıkışı, sola kaydırılması sırasında ise aynı MUX'un girişine  $A_3$  çıkışı uygulanır.

$S_1$	$S_0$	Hızlı bellek operasyonu
0	0	Değişiklik yok (aynı çıkış değer)
0	1	Sağa (aşağı) kaydırma
1	0	Sola (yukarı) kaydırma
1	1	Paralel yükleme



### 3.7 İkili Sayıcı

İkili sayıcı, önceden belirlenmiş durum dizisini izleyen sayısal devre olarak tanımlanır. Örneğin, 3-bit ikili sayıcı 3 FF'den oluşup, 000, 001, 010, 011, 100, 101, 110, 111 dizisini izler. Sayıcı genel olarak, komplement yapabilen FF' lerden oluşur. Sayıcılar, saat darbeli ardışıl sayısal devre yöntemleri ile tasarlanabilirler. Aşağıda 2-bit eşzamanlı ikili sayıcı görülmektedir.



Yukarıdaki örnekte JK FF' leri kullanılmıştır. Aynı sayıcı paralel load ve eşzamanlı silme (clear) eklenerek tasarlanırsa, JK FF' lerin giriş eşdeğeri aşağıda gösterildiği gibi olur. Alıştırma olarak aşağıdaki şekilde FF giriş eşitlikleri yazılabilir. Örneğin,  $J_0$  girişi için:  $J_0 = I_0 \cdot (\text{LOAD} \cdot \text{Clear}') + \text{INC} \cdot \text{Load}' \cdot \text{Clear}'$  yazılabilir.

### 3.8 Ana Bellek Ünitesi

Ana bellek ünitesi, bellek hücreleri ve ikili bilgiyi okumak ve yazmak için kullanılan devrelerden oluşur. İkili bilgiyi, byte (8 bit) ve kelime (2 Byte, 4 Byte, vb.) formatında saklamak mümkündür. Örneğin, 16 bitlik kelime formatı 2 Byte' lık kelimeyi tanımlar.

Ana bellekteki kelimelere adres bilgisi ile ulaşmak mümkün olur. Adres bilgisi ise adres hatları ile belleğe taşınır. k adres hattı ile 0-2k kelimeye erişmek mümkün olur. Bellek kapasitesi, byte sayısı ile belirlenir. Kullanılan birimleri K:kilo, M:mega ve G: gigadır.

$K=2^{10}$ ,  $M=2^{20}$  ve  $G=2^{30}$  byte kapasitesine karşı gelir. Örneğin, 8G,  $2^{33}$  byte kapasitesine karşı gelir.

### 3.9 Olasıl (Rastgele) Erişimli Bellek (RAM)

Olasıl erişimli belleğin (RAM) istenilen kelimesine olası olarak ulaşılabilir, okuma ve yazma yapılabilir. Aşağıda RAM'in blok diyagramı gösterilmiştir. Veri giriş ve çıkış hatları

yanında okuma ve yazma girişleri ve adres hatları vardır. Açıklandığı gibi,  $k$  adres hattı  $2^k$  kelimenin seçiminde kullanılabilir. Okuma işlemi, adres hatlarına gerekli ikili adres bilgisini gönderdikten sonra, oku girişini aktive ederek yapılır. Yazma işleminde ise, adres bilgisi adres hatlarına, giriş bilgisi giriş hatlarına gönderilir, sonra yaz hattı aktive edilir. Olasıl erişimli belleğin (RAM) blok diyagramı aşağıdaki gibidir

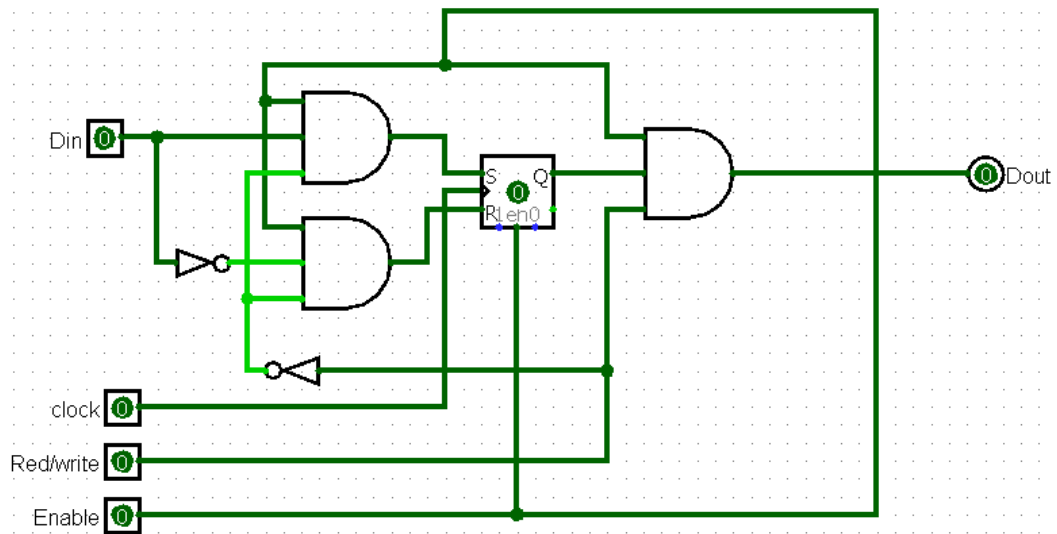


### 3.10 Yalnızca Okunabilir Bellek (ROM)

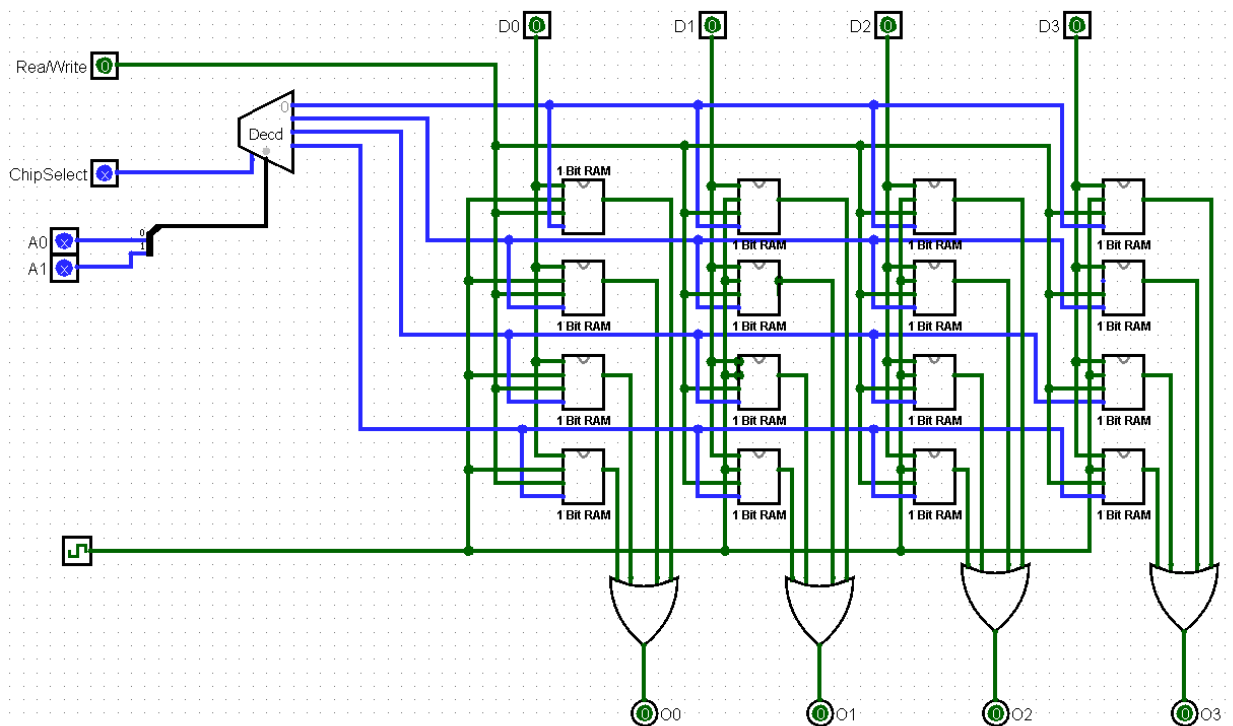
Yalnızca okunabilen bellekteki (ROM) ikili bilgi okunabilir ama değiştirilemez. RAM da bilgi istenildiğinde değiştirilebilir. İki tip belleğin kullanım alanları bu nedenle fark eder: RAM dinamik bellek olarak kullanılır. ROM ise ikili bilginin nadiren değiştirildiği uygulamalarda kullanılır. Örneğin, mikro program kontrolünde mikroprogram belleği olarak ROM kullanılır. ROM'un yapısı, yalnızca okunduğu varsayımıyla sadece okuma ve adres hattı ile veri çıkış hattından oluşur. Aşağıda yalnızca okunabilen belleğin (ROM) blok diyagramı gösterilmiştir.



### 1 Bit RAM



### 16 Bit (4 X 4 Bit) RAM



## 4 BÖLÜM YAZGAÇ –SAKLAYICI (Register) TRANSFER DİLİ ve MİKROİŞLEMLER

Bu bölümde, saklayıcılar arasındaki transfer işlemleri ve mikroişlemler görülecek, saklayıcı transfer dili ve notasyonu tanımlanacaktır. Hat grubu (bus) yoluyla veri transferleri incelenecektir. Aritmetik, lojik, kaydırma mikro işlemleri ve aritmetik, lojik, kaydırma birimleri tasarımı anlatılacaktır.

Sayısal bilgisayar bir işlevi yerine getirmek için çeşitli modüllerin bağlantılarına gereksinim duyar. Modüller, daha önce tanımlamış olduğumuz sayısal bileşenlerden oluşur. Bu modüller, en iyi olarak, saklayıcılar ve saklanan veri üzerindeki işlemlerle tanımlanabilir. Saklayıcılar üzerinde yapılan işlemlere mikroişlemler adı verilir. Bir sayısal bilgisayarın donanım yapısı en iyi aşağıdaki şekilde tanımlanabilir.

Saklayıcı transfer dili, saklayıcı bellek seviyesinde tüm aktiviteleri tam olarak tanımlar. Mikroişlemler hızlı bellek seviyesindeki işlemler olup, bu dil mikro işlemlerin gerçekleştirilmesinde kullanılır.

### 4.1 Saklayıcı (Register) Transfer Dili

Saklayıcı transfer dilinin temel sembolleri aşağıdaki gibi özetlenebilir.

Kullanılan sembol	Tanımı	Örnekler
Harfler	Saklayıcı Elemanı	R2, PC, IR
Parantez	Saklayıcıyı bir bölümü	R2 (8-15), R2(H)
Ok<=	Bilgi Transferi	R2<=R3
Virgül	İki mikroişlem	R2<=R3, R4<=R5

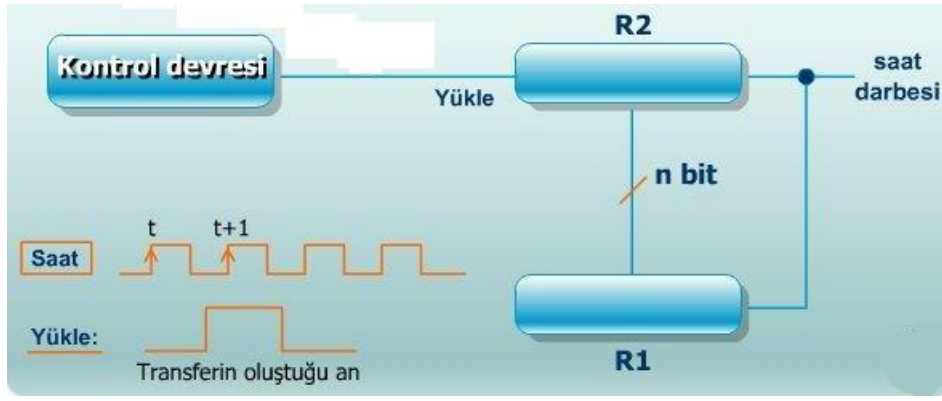
#### 4.1.1 R1 den R2 ye Bilgi Transferi

Bellek elemanı (Register) transfer dilinin ilk örneğini verip, detayları açıklayalım:

<b>Koşulsuz Saklayıcı Transferi</b>	<b>R2&lt;=R1</b>
-------------------------------------	------------------

Burada R1'ten R2'ye veri transferini gösterir. Saat darbesinin yükselen kısmında (kenar tetikleme)  $\text{yükle}=1$  göz önüne alınır ve R1'teki ikili bilgi R2'ye yüklenir.

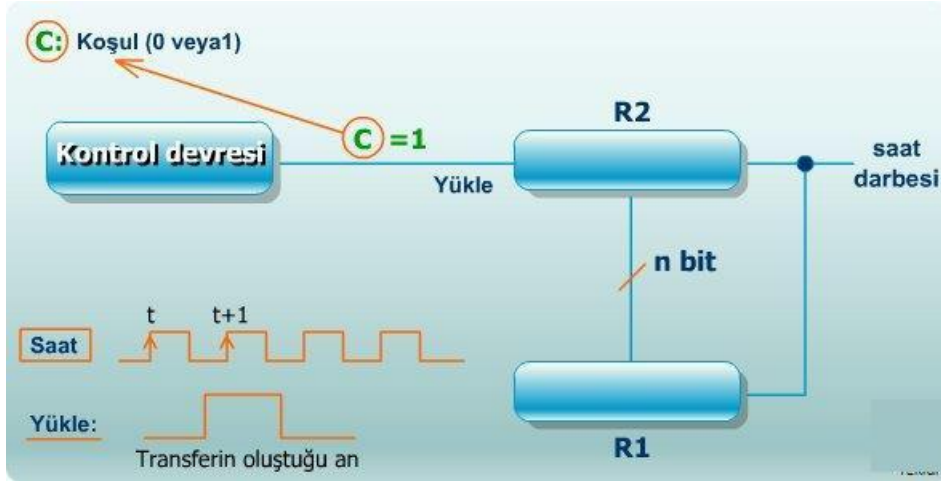
Bazı durumlarda bilgi transferi sadece belli koşul sağlandığında gerçekleşir



#### 4.1.2 R1 den R2 ye Koşullu Bilgi Transferi

**Koşullu Saklayıcı Transferi**  $C:R2 \leftarrow R1$

Burada C kontrol fonksiyonu veya koşulu,  $R2 \leftarrow R1$  ise R1'den R2'ye veri transferini gösterir. Kontrol koşulu iki nokta ile sonlandırılır. Koşul sağlanırsa (if-then), ardından gelen transfer gerçekleştirilir. Aşağıdaki şekilde C koşulu sağlandığında, yüklem girişi aktive olur (lojik 1). Saat darbesinin yükselen kısmında (kenar tetikleme)  $yüklem=1$  göz önüne alınır ve R1'teki ikili bilgi R2'ye yüklenir.

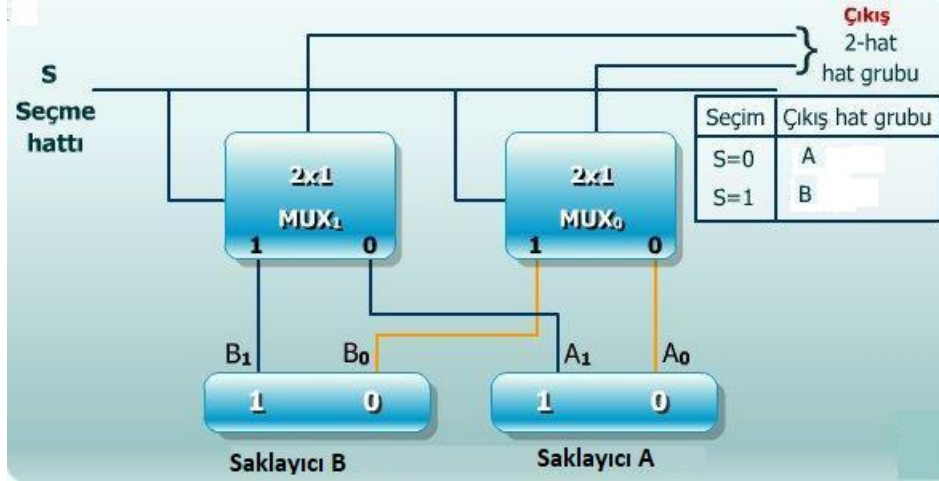


#### 4.2 Hat Grubu (Bus) Yoluyla Saklayıcılar Arası Veri Transferi

Tipik bir bilgisayarda saklayıcıları (register) bir grup hattın oluşan bağlantılarla birbirine bağlanmışlardır. 8 bitlik bir saklayıcı, diğer 8 bitlik saklayıcıya 8 ayrı tel ile bağlanmış olarak düşünülebilir. Bir elemanın her bir biti, diğer elemanında karşı düşen bite bağlanmış olarak düşünülebilir. Böylece elemanlar arasında hat grubu (bus) bağlantısı yapılmış olur. Bu kısımda genel hat grubu bağlantısı oluşturmanın iki yolundan bahsedeceğiz:

#### 4.2.1 MUX (Multiplexer Bus) Yoluyla Genel Hat Grubu (BUS) Bağlantısının Yapılması

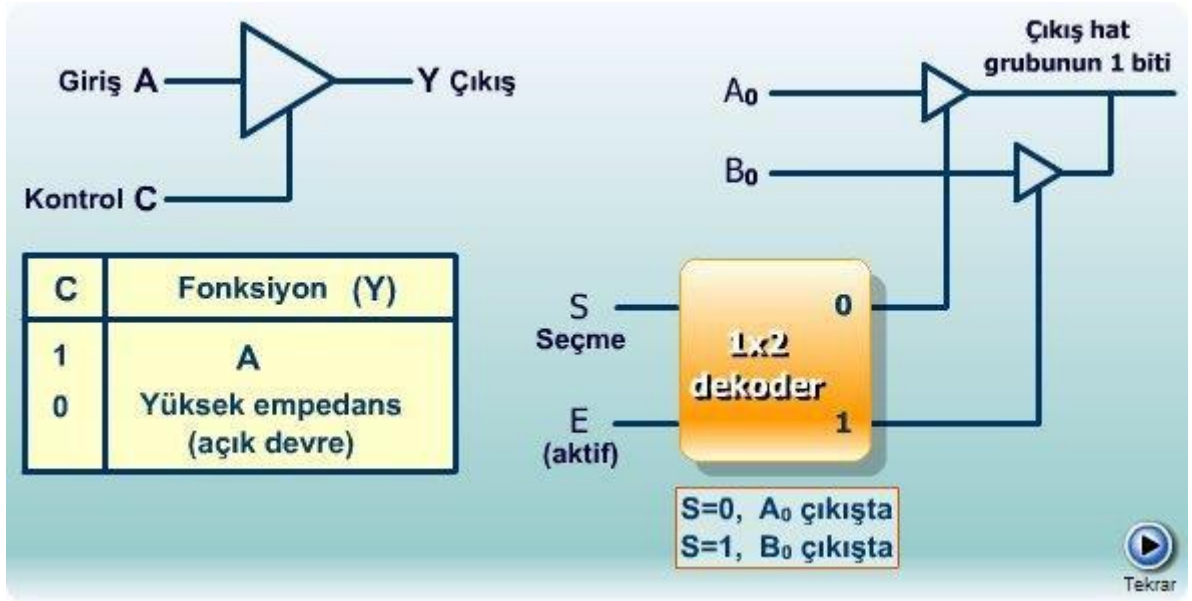
Aşağıda 2 bellek elemanının 2 MUX ile genel hat grubu bağlantısının yapılması açıklanmıştır. Girişteki belleklerden R1'in bitleri MUX'ların ilk bitlerine bağlanmış olup, seçim hattı S1'in seçilmesi ile genel hat grubuna ulaşırlar.

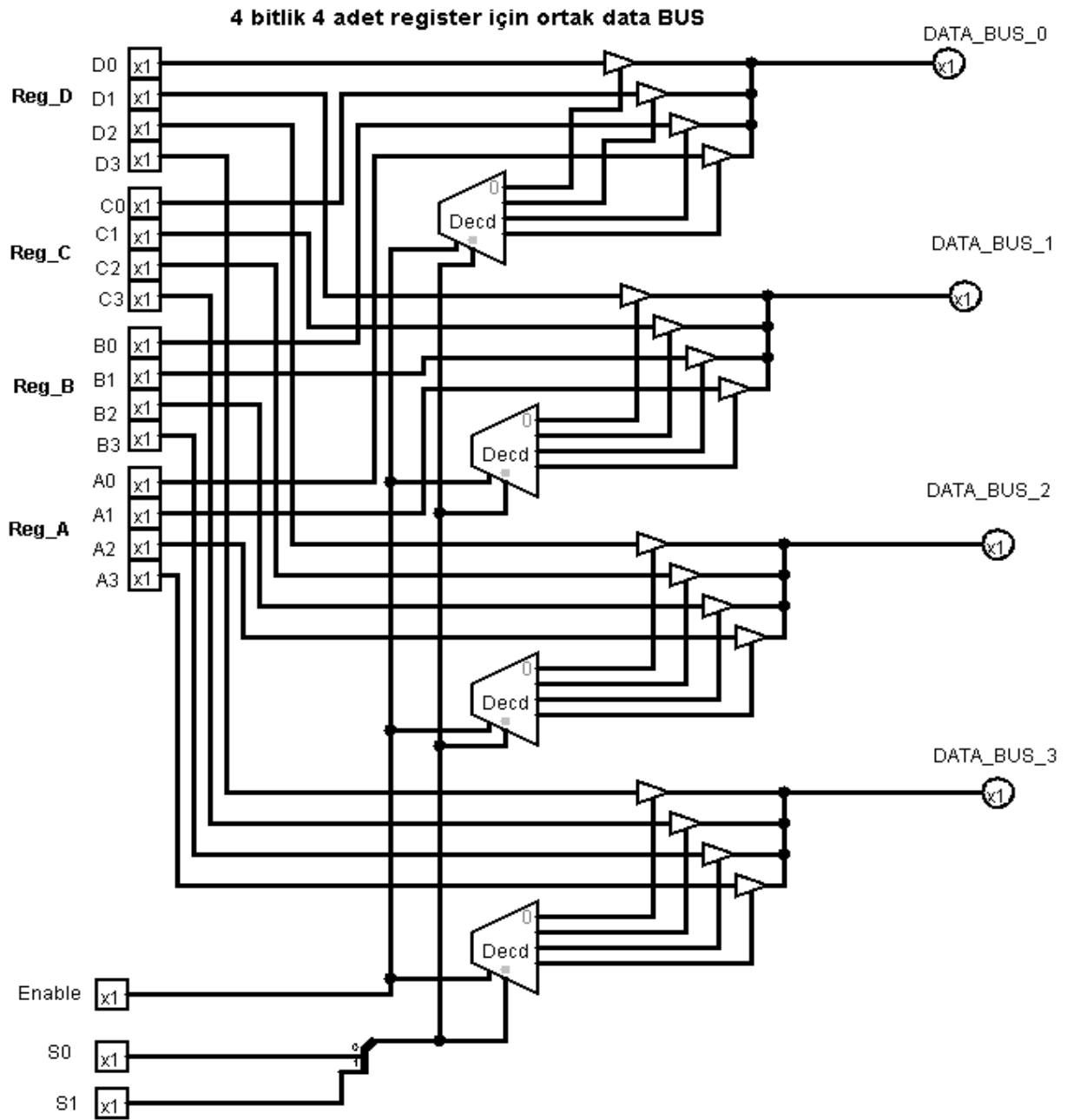


Burada 4 adet 4 bitlik saklayıcıyı ortak B u s ' a bağlamak için 4 adet 4x1 MUX gerekir. Bu MUX'ların 2 seçim hatları ortak olarak bağlanır.

#### 4.2.2 Üç-Durumlu Kapılar Yoluyla Genel Hat Grubu (BUS) Bağlantısının Yapılması

Üç-durumlu kapılar, ortak BUS bağlantılarında yaygın olarak kullanılan elemanlardır. İki durumları normal lojik devre ödevi görür. Eğer eleman bir üç-durumlu koruyucu (buffer) ise girişteki 0 ve 1 lojik seviyeler aynen çıkışa aktarılır. Üçüncü durumu ise "yüksek empedans" durumudur. Bu durumda giriş ve çıkış bağlantısı kesilir, izolasyon oluşur





### 4.2.3 Ana Bellek Bilgi Transferi

Ana bellek transfer işlemleri diğer önemli bilgi transfer işlemleridir. Bu işlemler Okuma ve yazma işlemleri olarak tanımlanabilir. Ana bellek işlemleri, AR (address register) ve DR (data register) saklayıcıları yoluyla yapılır.

<b>Okuma</b>	$DR \leftarrow M[AR]$
--------------	-----------------------



**Yazma** $M[AR] \leq R2$ 

AR ana belleğin adresini tutar, DR ise bu adrese yazılacak veya okunacak ikili bilgiyi tutar.

### 4.3 Aritmetik Mikroişlemler

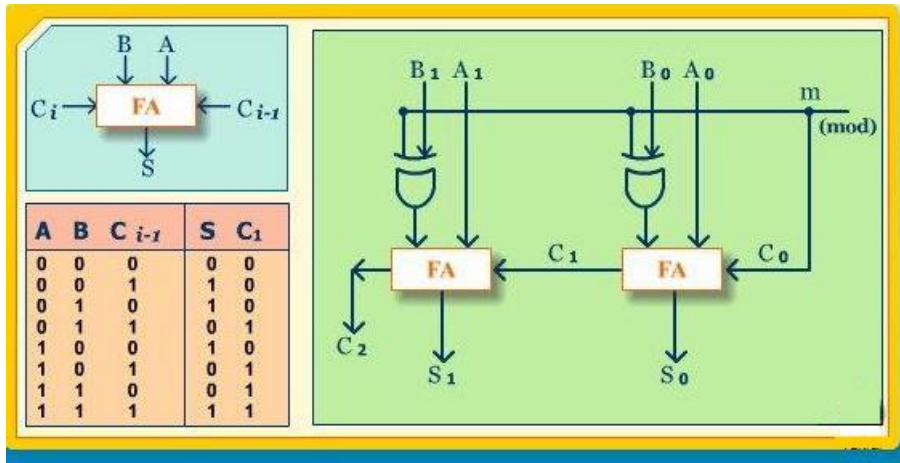
Aritmetik mikroişlemler, Aritmetik işlemlerin saklayıcılar yoluyla gerçekleşmesi olarak tanımlanabilir. Bu bölümde, sadece toplama ve çıkarma gibi temel işlemleri sayısal bileşenlerle gerçekleyeceğiz. Aritmetik mikroişlemler aşağıdaki tabloda gösterilmiştir:

$R4 \leq R2 + R3$	Toplama
$R4 \leq R2 - R3$	R2 eksi R3
$R4 \leq R4'$	Komplement (1-komplement)
$R4 \leq R4' + 1$	Negatif R4 (2-komplement)
$R4 \leq R2 + R3' + 1$	Çıkarma (Tümleme işlemi ile)
$R4 \leq R4 + 1$	Arttırma
$R4 \leq R4 - 1$	Eksiltme

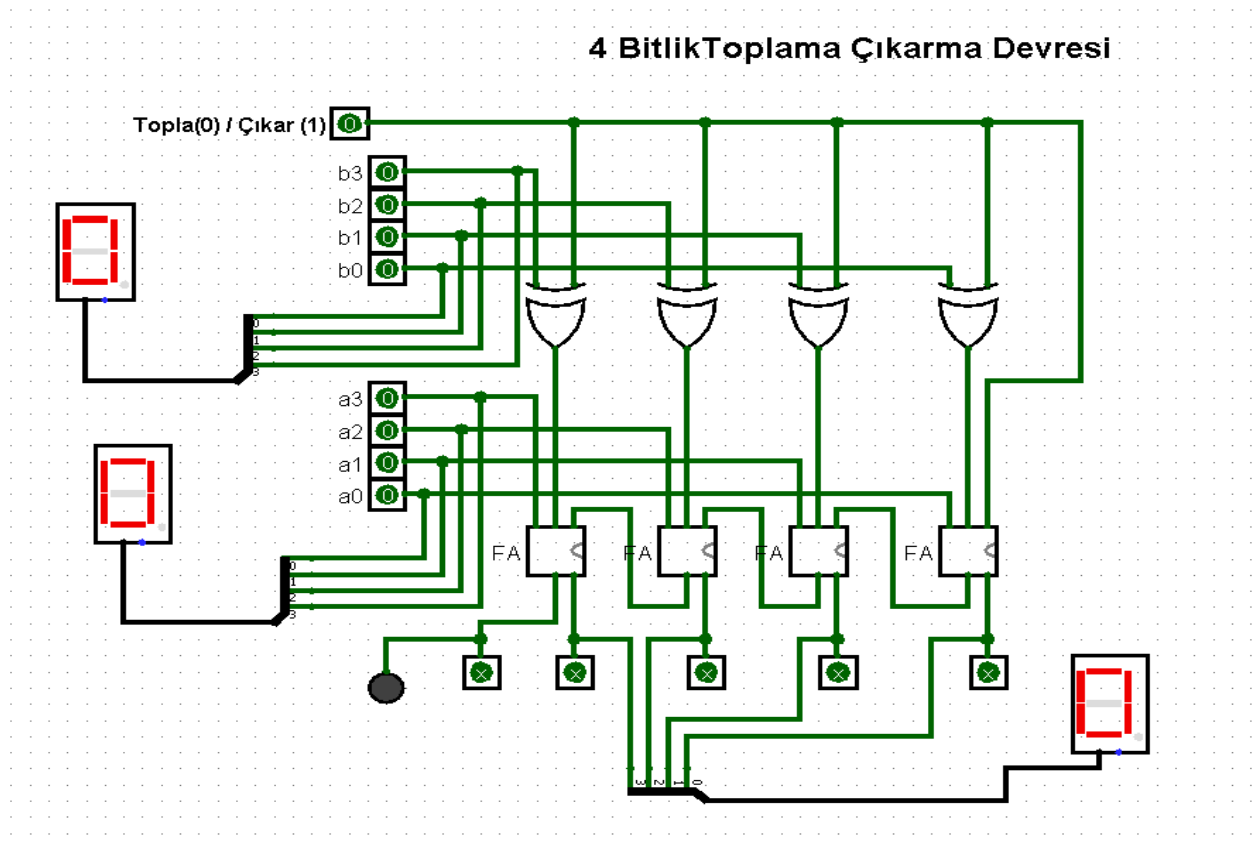
#### 4.3.1 2-Bitlik Toplama-Çıkarma İşlemi

Toplama işlemi yapmak için tam toplayıcı (full adder-FA) kullanılır. FA, elde (carry) girişi kullanarak ikili sayıları topladığı için 1-bitlik toplama işlemi yapmak için uygun olur.

2 ikili sayının çıkarma işlemi, bir sayı ile diğer sayının 2-tümleyeninin toplanması olarak gerçekleştirilebilir. Bu durumda toplama ve çıkarma için tek bir devre kullanılabilir. Aşağıdaki şekilde FA devresinin blok diyagramı ve 2-bitlik toplama-çıkarma devresinin blok diyagramı gösterilmiştir.

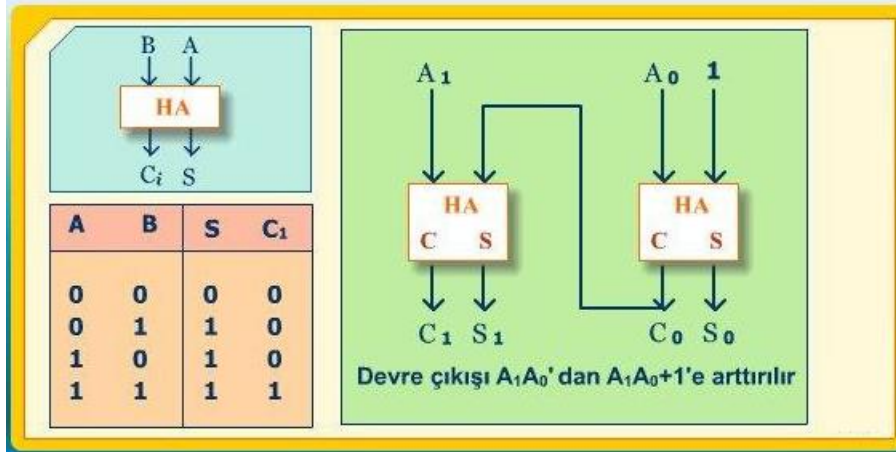


FA devresinin 2 girişinden  $A_i$  ve  $B_i$ 'yi, elde  $C_{i-1}$  ile toplayarak  $S_i$  toplamını ve  $C_i$  eldesini elde etmek mümkün olur. N-bitlik toplama devresi hücreleri yan yana bağlayarak elde edilir. Aynı devreye, çıkarma işlemini eklemek için  $M \pmod{2}$  girişi kullanılır.  $M=0$  olduğunda devre toplama işlemi yapar.  $M=1$  olduğunda  $B$  'nin 2-tümleyeni üretilir ve devre çıkarma işlemi yapar.



### 4.3.2 2-Bitlik Arttırma İşlemi

İkili arttırma (binary incrementer), saklayıcının içeriğini 1 arttırmaktır. Bu mikroişlem ikili sayıcı ile kolaylıkla gerçekleştirilebilir. Bu durumda bir saat darbesi gecikme yaratır. Bir diğer gerçekleştirme yöntemi ise Yarım Toplayıcı (Half Adder- HA) devresi kullanmaktır. Aşağıda 2-bitlik arttırma işlemi 2 HA devresi ile gerçekleştirilmiştir.



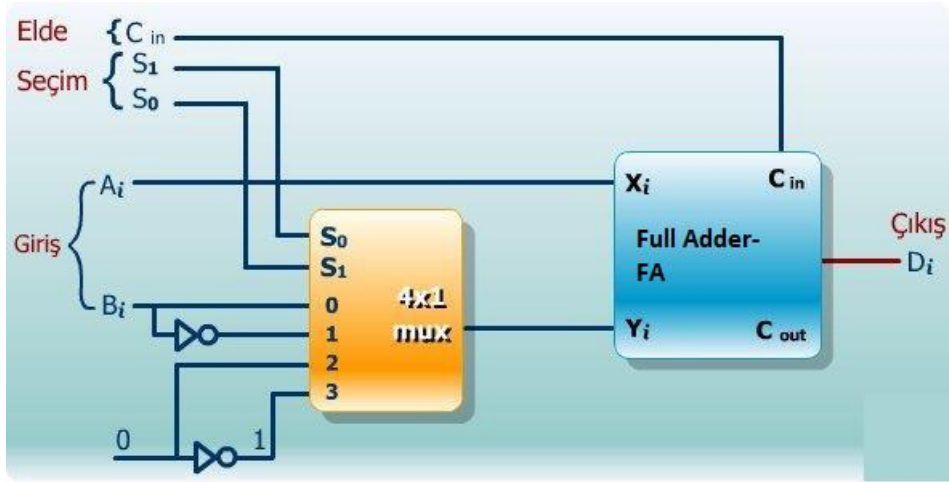
### 4.3.3 1-Bitlik Aritmetik Devre Tasarımı

Bu bölümün başında tanımlanan aritmetik mikroişlemleri yapabilen 1-bitlik bir devre tasarımı aşağıda gösterilmiştir.

Devre bir FA ve bir 4x1 MUX devresinden oluşmaktadır. Aritmetik devre çıkışı aşağıdaki toplam yoluyla oluşturulur:

$$D_i = A_i + Y_i + C_{in}$$

Girişler  $A_i$ ,  $B_i$  ve elde  $C_{in}$ , MUX'un seçme hatları  $S_1$  ve  $S_0$  yoluyla aritmetik mikroişlemler tablosunda tanımlanan işlemlerin gerçekleştirilmesini sağlar.



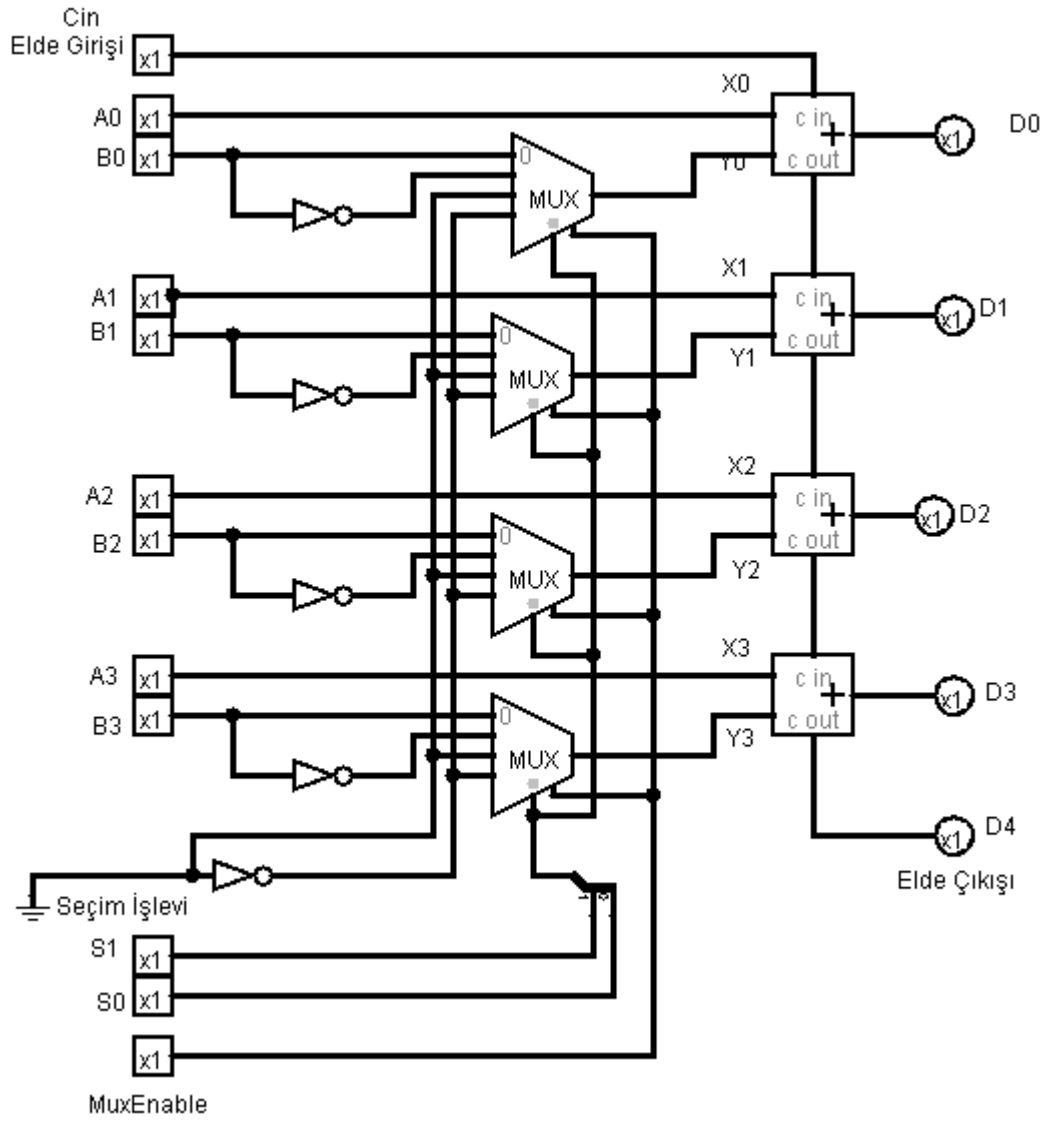
#### 4.3.4 Aritmetik Mikroişlemciler Tablosunda Aritmetik Devrenin Gerçekleştirdiği İşlemler

S1	S0	Ci	Yi	$D_i = A_i + Y_i + C_{in}$	Mikroişlem
0	0	0	B	$D = A + B$	Toplama
0	0	1	B	$D = A + B + 1$	Eldeli Toplama
0	1	0	$B'$	$D = A + B'$	Borçlu çıkar
0	1	1	$B'$	$D = A + B' + 1$	Çıkar
1	0	0	0	$D = A$	Transfer A
1	0	1	0	$D = A + 1$	A yı 1 arttır
1	1	0	1	$D = A - 1$	A yı 1 azalt
1	1	1	1	$D = A$	Transfer A

n bitlik aritmetik devre tasarımı, yukarıda açıklanan 1 bitlik aritmetik devrelerin birbirine paralel olarak bağlanmasıyla sağlanır:  $C_{in}$ ,  $S_1$ ,  $S_0$  girişleri tüm bitler için ortak olacaktır.

Ayrıca komşu bitlerin  $C_{in}$  ve  $C_{out}$ 'ları zincirleme olarak birbirine bağlanacaktır. En anlamlı bitin  $C_{out}$ 'u tüm devrenin  $C_{out}$ 'u olarak tanımlanacaktır.

## 4 bitlik Aritmetik İşlem Devresi



#### 4.4 Lojik (Mantıksal) Mikroişlemler

Lojik mikroişlemler saklayıcı elemanlarında depolanan bit dizilerine uygulanan lojik operasyonları tanımlar. Bu operasyonlar her bit için bağımsız olup, her bir bite “bit bazında” uygulanır. Aritmetik mikroişlemlerde elde ve ödünç bitleri ile bir sonraki bite gönderilen bilgi bu işlemlerde bulunmaz. Saklayıcı transfer dilinde tanımladığımız AND lojik işlemi  $\wedge$  sembolü, OR lojik işlemi ise  $\vee$  sembolü ile gösterilir. Bazı uygulamalarda OR işlemi  $+$  sembolü ile gösterilebilir.

$R2 \leftarrow R2 \wedge R3, R7 \leftarrow R2 \vee R5$   
 $R2 \leftarrow R2 + R3, R7 \leftarrow R2 \vee R5$

Aşağıdaki örnekte ise belli bir koşul sağlandığında ( $P \wedge Q$ ) meydana gelen saklayıcı aktarım fonksiyonu verilmiştir.

$P \wedge Q : R1 \leftarrow R2 + R3$

İki ikili değişken ile tanımlanabilen 16 adet lojik fonksiyon tanımlanabilir. Aşağıdaki tabloda 16 farklı logic fonksiyonun x ve y giriş değerlerinde aldığı değerler verilmiştir.

x	y	F <sub>0</sub>	F <sub>1</sub>	F <sub>2</sub>	F <sub>3</sub>	F <sub>4</sub>	F <sub>5</sub>	F <sub>6</sub>	F <sub>7</sub>	F <sub>8</sub>	F <sub>9</sub>	F <sub>10</sub>	F <sub>11</sub>	F <sub>12</sub>	F <sub>13</sub>	F <sub>14</sub>	F <sub>15</sub>
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

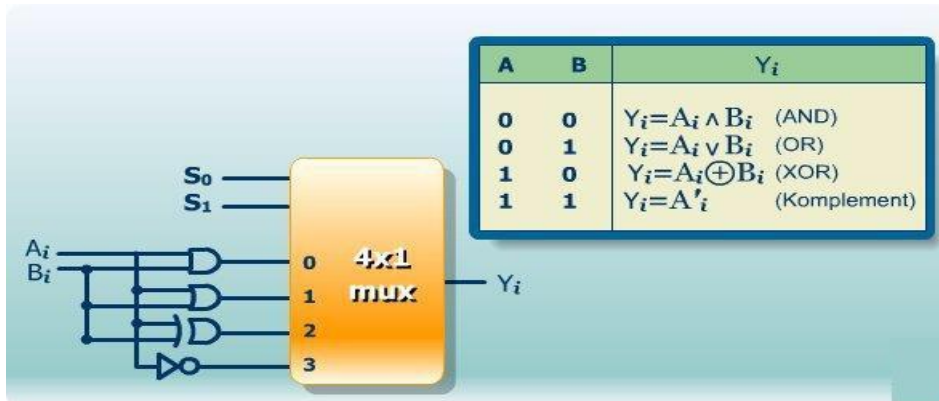
Bu logic 16 fonksiyon ise aşağıda tabloda verilmiştir.

Boolean Fonksiyonu	Mikroişlem	Açıklama
$F_0=0$	$F \leftarrow 0$	Silme
$F_1=xy$	$F \leftarrow A \wedge B$	AND (VE)
$F_2=xy'$	$F \leftarrow A \wedge B'$	
$F_3=x$	$F \leftarrow A$	A'nın aktarımı
$F_4=x'y$	$F \leftarrow A' \wedge B$	

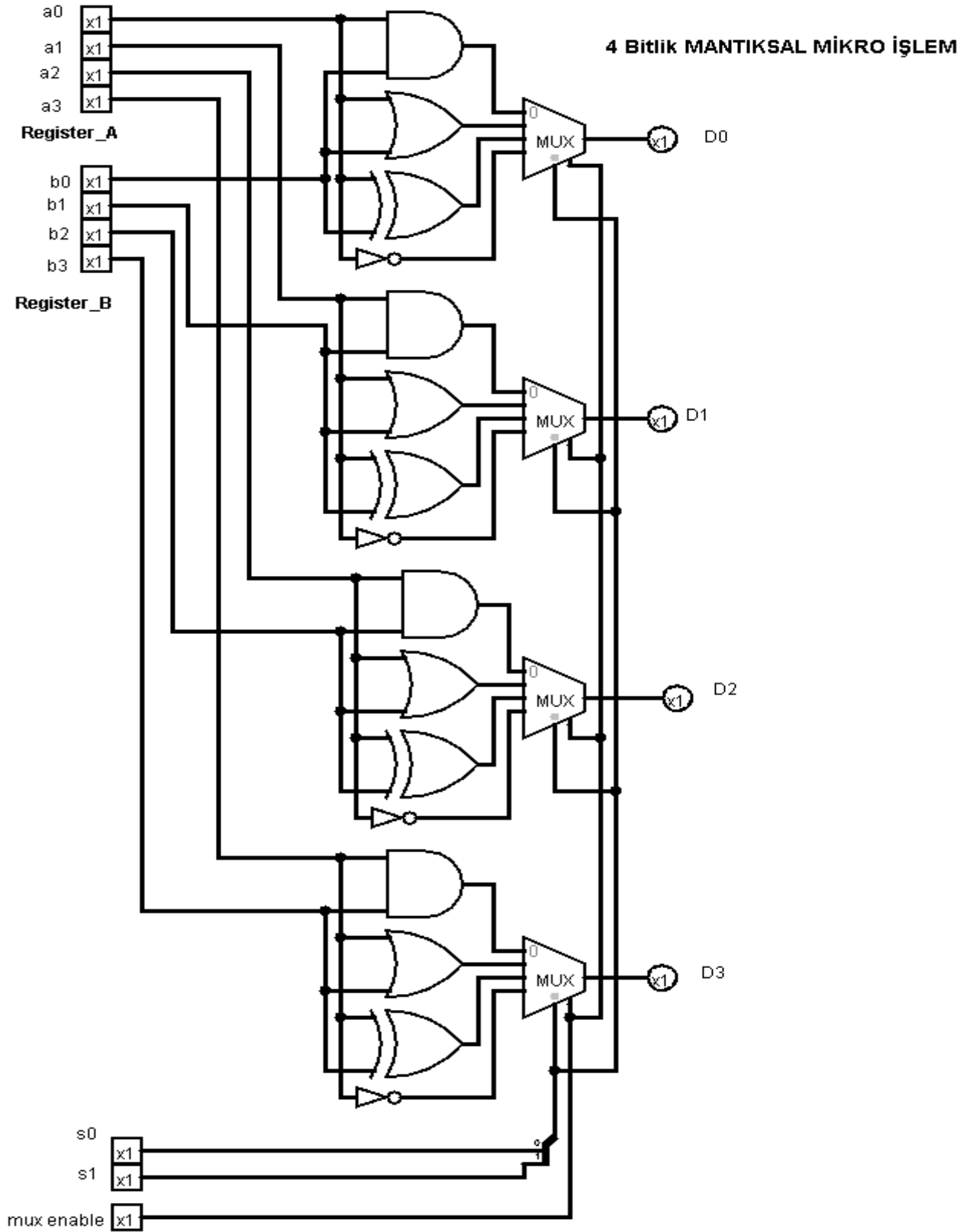
$F_5=y$	$F \leftarrow B$	B'nin aktarımı
$F_6=x \oplus y$	$F \leftarrow A \oplus B$	EXOR (ÖZEL VEYA)
$F_7=x+y$	$F \leftarrow A \vee B$	OR (VEYA)
$F_8=(x+y)'$	$F \leftarrow (A \vee B)'$	NOR (VEYA DEĞİL)
$F_9=(x \oplus y)'$	$F \leftarrow (A \oplus B)$	EXNOR (ÖZEL VEYA DEĞİL)
$F_{10}=y'$	$F \leftarrow B'$	B'nin tümleyeni
$F_{11}=x+y'$	$F \leftarrow A \vee B'$	
$F_{12}=x'$	$F \leftarrow A'$	A'nın tümleyeni
$F_{13}=x'+y$	$F \leftarrow A' \vee B$	
$F_{14}=(xy)'$	$F \leftarrow (A \wedge B)'$	NAND (VE DEĞİL)
$F_{15}=1$	$F \leftarrow 1$	1 leme

#### 4.4 1-bitlik Lojik Mikroişlem Biriminin Gerçeklenmesi

Boole fonksiyonlarının bilinen bir diğer özelliği ise, sadece bazı temel fonksiyonlarla diğer bütün işlemlerin yapılabileceğidir: örneğin, sadece AND, OR, XOR ve NOT kullanılarak diğer işlemler gerçekleştirilebilir. Aşağıda 1-bitlik lojik mikroişlem biriminin gerçekleştirilmesi gösterilmiştir.



Devrede 4x1 MUX'un girişlerindeki AND, OR, XOR ve NOT fonksiyonlar seçme hatları  $S_1$  ve  $S_0$  ile seçilerek çıkışa verilir. n bitlik Lojik devre için 1 bitlik her bir Lojik devreden n tanenin paralel olarak görülmesi gerekir. Her bir bitlik birim bağımsız bir çıkış üretecektir. Diğer taraftan, 1-bitlik Lojik devrede üretilen işlemler AND, OR, XOR ve NOT, diğer tüm Lojik işlemlerin (fonksiyonlarının) Boole cebri vasıtasıyla elde edilmesini sağlar.



#### 4.5 Kaydırma (Shift ve Rotate) Mikroişlemleri

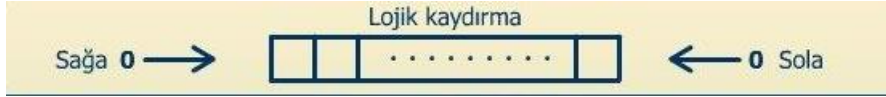
Kaydırma mikroişlemleri seri veri transferi ve diğer mikroişlemlerle birlikte, çarpma, bölme gibi çeşitli operasyon gerçekleştirmek için kullanılırlar.



Üç tip kaydırma işlemi vardır. Bunlar Mantıksal Kaydırma, Dairesel Kaydırma (Rotate), Aritmetik Kaydırma olarak tanımlanabilir. Bu kaydırma işlemlerinde saklayıcı içerikleri Sağ veya Sol yönde kaydırılabilir.

#### 4.5.1 Lojik Kaydırma

Lojik kaydırmada, seri girişten 0 transfer edilerek, saklayıcı içeriği sağa ve sola kaydırılır. Bu sırada, içeriğin karşı uçtan en son biti kaybolur.



#### 4.5.2 Dairesel Kaydırma

Dairesel kaydırmada ise karşı uçtan gelen bit, seri giriş olarak kaydırılan uca uygulanır. Böylece, bellek içeriği dairesel olarak hareket etmiş olur.



#### 4.5.3 Aritmetik Kaydırma

Aritmetik kaydırmada, saklayıcı içeriğine seri girişten 0 uygulanması ve işaretinin değişmemesi önemlidir. Aritmetik kaydırmada işaret biti olduğu gibi kalır değişmez. Dolayısıyla, en soldaki işaret biti sağ ve sola kaydırmada aynı kalmalıdır. Aritmetik sola kaydırma işaretli ikili sayıyı 2 ile çarpmak, sağa kaydırma ise 2 ile bölmektir.

- Aritmetik Sağa kaydırmada  $R_{n-1}$ ,  $R_{n-2}$ 'nin yerine geçer ve tüm bitler sağa kaydırılır.  $R_0$  bitinin içeriği kaybolur.
- Sola kaydırmada ise  $R_0$  içeriğine sıfır gelir ve tüm bitler sola kaydırılır.  $R_{n-1}$  biti kaybolarak  $R_{n-2}$ ,  $R_{n-1}$  yerine geçer. Kaydırma işlemi sonunda  $R_{n-1}$  in değeri değişmiş ise tersine çevrilir.

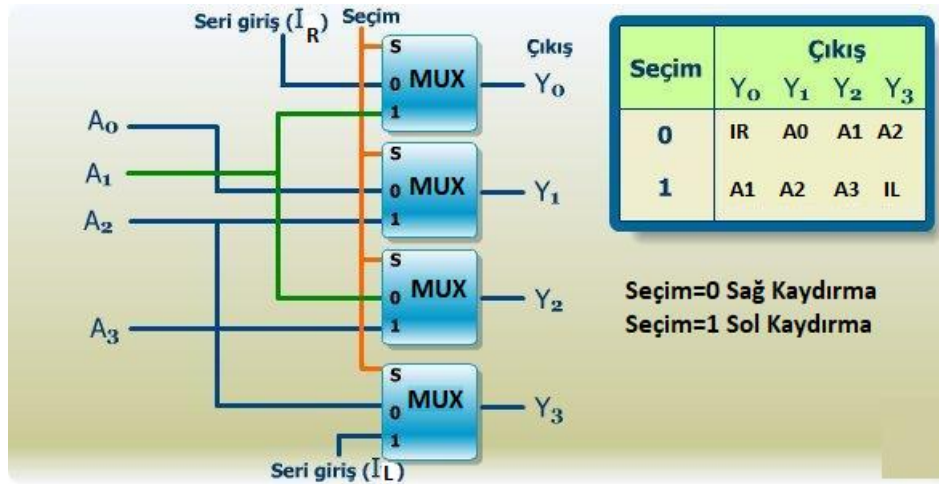


## Kaydırma Mikro İşlemleri

Sembolik	Açıklama
$R \leftarrow \text{SHL } R$	R Saklayıcısının Mantıksal Sola Kaydırılması
$R \leftarrow \text{SHR } R$	R Saklayıcısının Mantıksal Sağa Kaydırılması
$R \leftarrow \text{CIL } R$	R Saklayıcısının Dairesel Sola Kaydırılması
$R \leftarrow \text{CIR } R$	R Saklayıcısının Dairesel Sağa Kaydırılması
$R \leftarrow \text{ASHL } R$	R Saklayıcısının Aritmetik Sola Kaydırılması
$R \leftarrow \text{ASHR } R$	R Saklayıcısının Aritmetik Sağa Kaydırılması

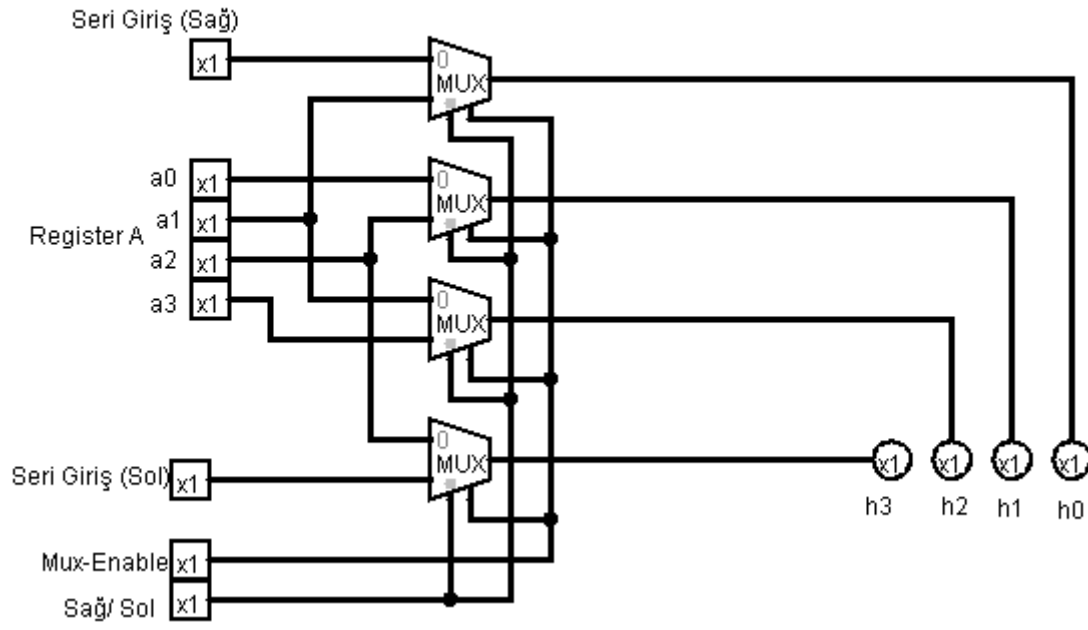
### 4.5.4 Kaydırma Yapan Saklayıcının Tasarımı

Kaydırma yapan saklayıcının tasarımında iki yoldan bahsedebiliriz: Birincisinde, daha önce açıklanan iki-yönlü kaydırmalı paralel yüklemeli saklayıcı kullanmak mümkündür. İkincisinde ise aşağıda gösterilen 4-bitlik MUX larla yapılan devre kullanılabilir



Bu devrede kullanılan dört 2x1 MUX'un uygun girişlerini seçerek, 1 saat darbesi gecikme olmaksızın, iki-yönlü kaydırma işlemi gerçekleştirilir.

#### 4 bitlik Mantıksal Kaydırma Devresi

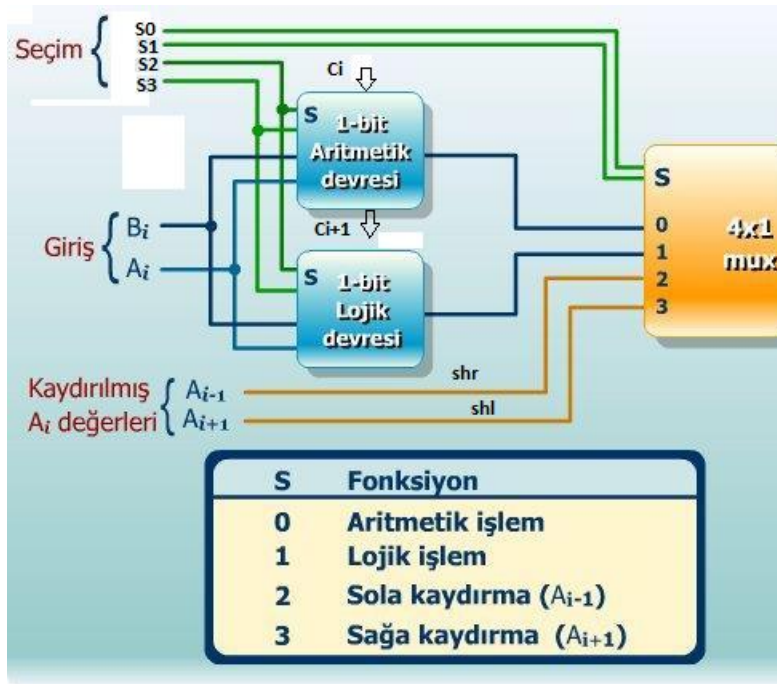


## 4.6 Aritmetik, Lojik, Kaydırma (Alu) Birimi

Bir bilgisayarda, tüm operasyonların gerçekleşmesi için aritmetik, lojik ve kaydırma işlemlerini yapan devrelerin, veri depolayan saklayıcılar ile birleştirilerek ALU oluşturulması gerekir. Aşağıda her bir işlemi yapan 1-bitlik blokların bir MUX ile birleştirilerek tüm işlemleri yapan bir devre elde edilebileceği gösterilmiştir.

$S_1 S_0$  seçim hatları yoluyla aritmetik ve lojik işlem çeşitleri,  $S_2 S_3$  hatları yoluyla çıkış MUX'dan kaydırma dahil herhangi bir işlem seçimi mümkün olmaktadır. 1-bitlik ALU devresine dikkat edilirse çıkış MUX'da tüm işlemlerin 1 bit bazında elde edilmesi mümkündür.

Aritmetik ve Lojik devre blokları gerçekleşip, çoğaltılarak istenilen n bit ALU devre tasarımı elde edilebilir.



Böyle bir devre ile elde edilebilecek mantıksal , matematiksel ve kaydırma işlevleri aşağıdaki tabloda verilmiştir.

S <sub>3</sub>	S <sub>2</sub>	S <sub>1</sub>	S <sub>0</sub>	C <sub>i</sub>	İşlem	Açıklama
0	0	0	0	0	$F=A$	A nın aktarımı
0	0	0	0	1	$F=A+1$	A nın değerini 1 arttır
0	0	0	1	0	$F=A+B$	Toplama
0	0	0	1	1	$F=A+B+1$	Eldeli Toplama
0	0	1	0	0	$F=A+B'$	Ödünç ile Çıkarma

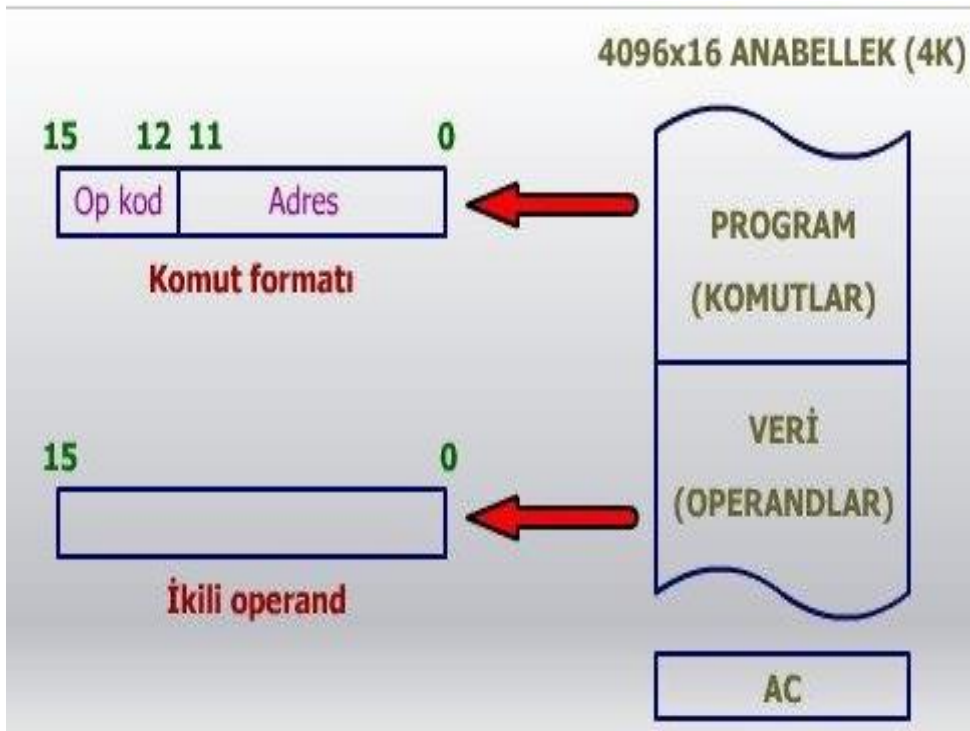
0	0	1	0	1	$F=A+B'+1$	Çıkarma
0	0	1	1	0	$F=A-1$	A'nın değerini 1 azalt
0	0	1	1	1	$F=A$	A'nın aktarımı
0	1	0	0	x	$F=A \wedge B$	VE işlemi
0	1	0	1	x	$F=A \vee B$	VEYA işlemi
0	1	1	0	x	$F=A \oplus B$	ÖZEL VEYA işlemi
0	1	1	1	x	$F=A'$	A'nın Tümlenyeni
1	0	x	x	x	$F=\text{shr } A$	A'nın Mantıksal Sağ kaydırımı
1	1	x	x	x	$F=\text{shlr } A$	A'nın Mantıksal Sol kaydırımı

## 5 TEMEL SAYISAL BİLGİSAYARIN KOMUT ve SAKLAYICI (REGISTER) YAPISI

Bir sayısal bilgisayarın, kullanıcı tarafından kullanımı “*program*” ve verilen “*veri*” ile mümkün olur. Program çeşitli görevleri yapan *komutlardan* oluşur. Veri-işleme görevi, yeni bir programın yazılımı ve aynı ve farklı veri üzerinde koşturulması olarak tanımlanabilir. Bir bilgisayar komutu belli bir işlemi gerçeklemek için tasarlanmış bir grup bittir. Bir komut kodu iki parçadan oluşmuştur: **operasyon kısmı(opcode)** ve **adres kısmı (operand)**.

- Operasyon (**opcode**) kısmı yapılan işlemin tanımlandığı bir grubudur.
- Adres kısmı (**operand**) ise, tanımlanan moda göre, operand adresinin tanımlandığı bitlerdir. Operand hesaplanan “efektif adres” değerine göre bellekten okunur.

En kullanışlı bilgisayar yapılarından biri, işlemciye ait bir saklayıcı (*register*) elemanı (akümülatör-AC) kullanmaktır. İşlem görececek operandı ve işlem sonucunu burada saklamaktır.

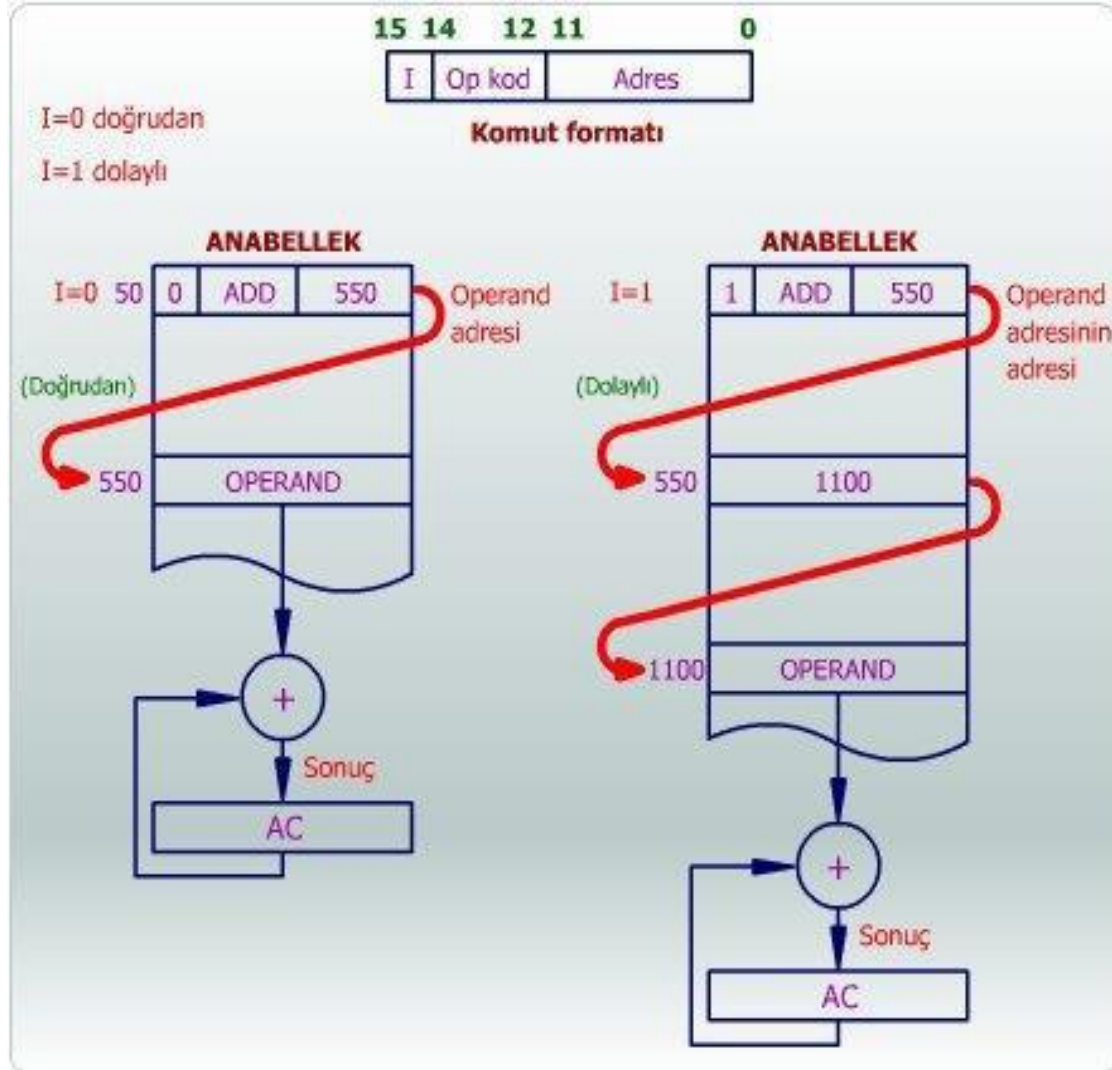


Her bir bilgisayar komutu, bir dizi mikro işlem ile gerçekleştirilir ve her bilgisayarın kendine özgü bir komut seti vardır. Bilgisayar komutları, ana bellekte ardarda gelen adreslerde saklanır ve sıra ile koşturulur. Bir komut koşturulurken, arkasından gelen komutun adresi hesaplanır. Kontrol birimi, komutu oluşturan mikroişlemleri ardışık gelen saat darbelerinde gerçekleştirir.

Bir bilgisayarın komutu koşması sırası ile;

1. Komutu bellekten okur
2. Saklayıcıya yerleştirir
3. Kontrol modülü komutun opcode bilgisini yorumlar
4. Komut icra edilir

## 5.1 Bilgisayarın Adresleme Modları

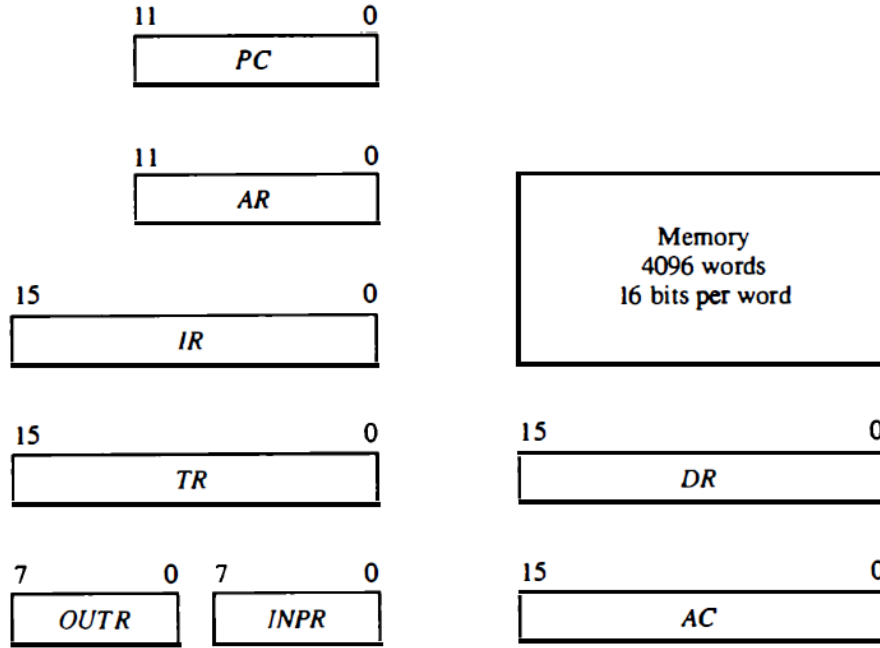


Adresleme modları olarak, operandı içeren *derhal* (*immediate*), *doğrudan* (*direct*) ve *dolaylı* (*indirect*) modlar kullanılmıştır. Efektif adres, adres modu tarafından hesaplanan adres olarak tanımlanır.

- **Derhal modunda**, komutun adres kısmı komutun kullanacağı veriyi gösterir.
- **Doğrudan modda**, komutun adres kısmı operandın yerini gösterir.
- **Dolaylı adres modunda**, komutun adres kısmı “adresin saklandığı yerin” veya adresin adresinin değeridir.

## 5.2 Bilgisayarın Saklayıcı Yapısı

Mikroişlemlerin 16 bit olarak tanımlanan bir komut formatının 4 bitlik kısmının operasyon kodu (**opcode**) olduğu hatırlanırsa, geri kalan 12 bitlik adres bilgisinin 4096 ( $2^{16}=4096$ ) kelimelik bir anabellek değerini adresleyebileceği açıktır.



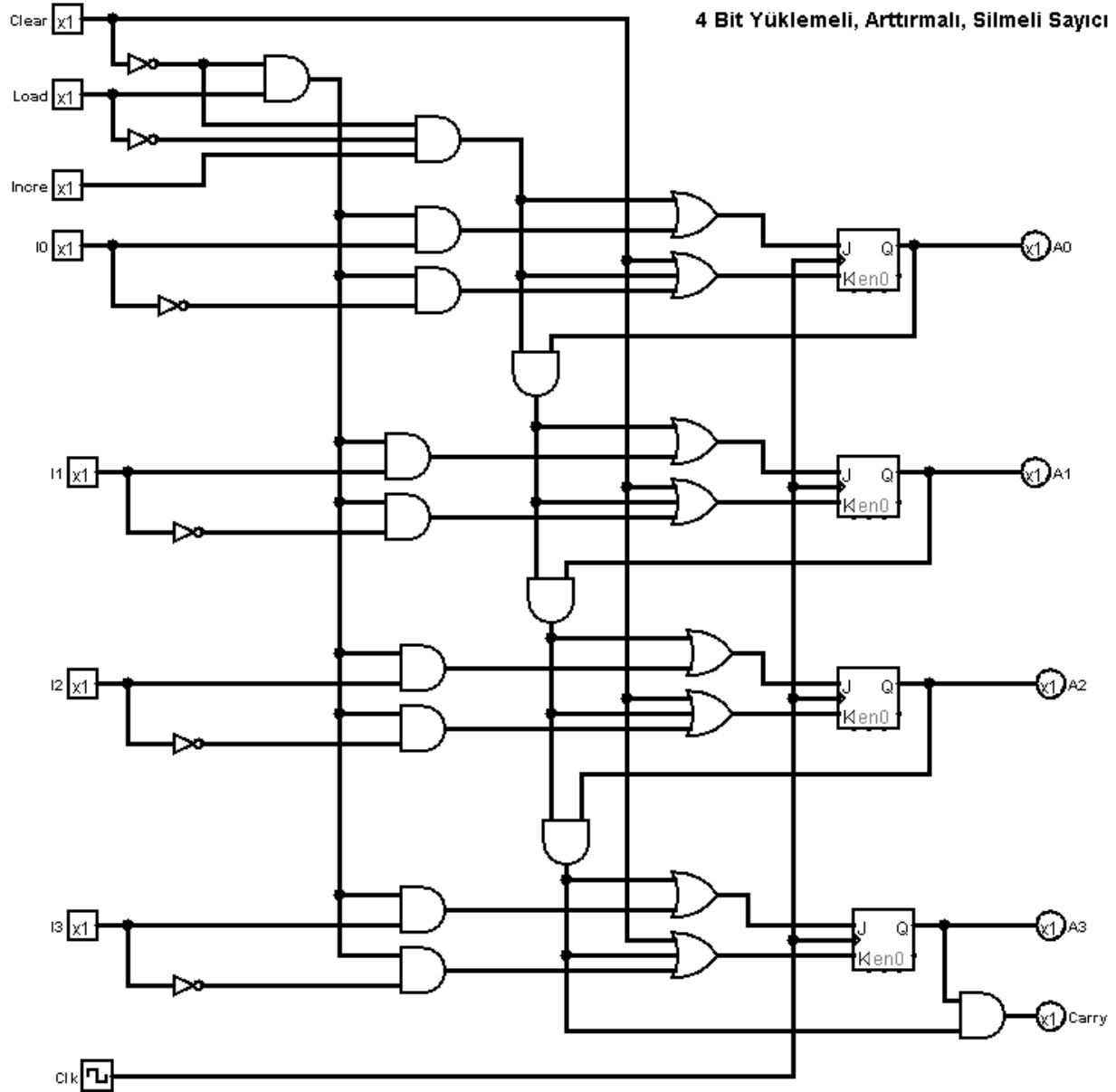
Temel bilgisayar yapısında aşağıdaki tablodaki saklayıcılar (register) kullanılmaktadır:

Saklayıcı Sembolü	Büyük­lük (bit)	Saklayıcı	Amaç
DR	16	Veri Saklayıcı	Ana bellek okuma ve yazma işlemlerinde veri bilgisini tutar.
AR	12	Adres Saklayıcı	Ana bellek okuma ve yazma işlemlerinde adres bilgisini tutar.
AC	16	Akümülatör	Yapılacak olan matematiksel ve mantıksal işlemlerin operantlarını ve sonucunu saklar.
IR	16	Komut Saklayıcı	Komutun saklanması, operasyonunun yorumlanması, mikroişlemlerin başlatılması görevini yapar.
PC	12	Program Sayacı	Koşulan komuttan sonraki komutun adresini gösterir.

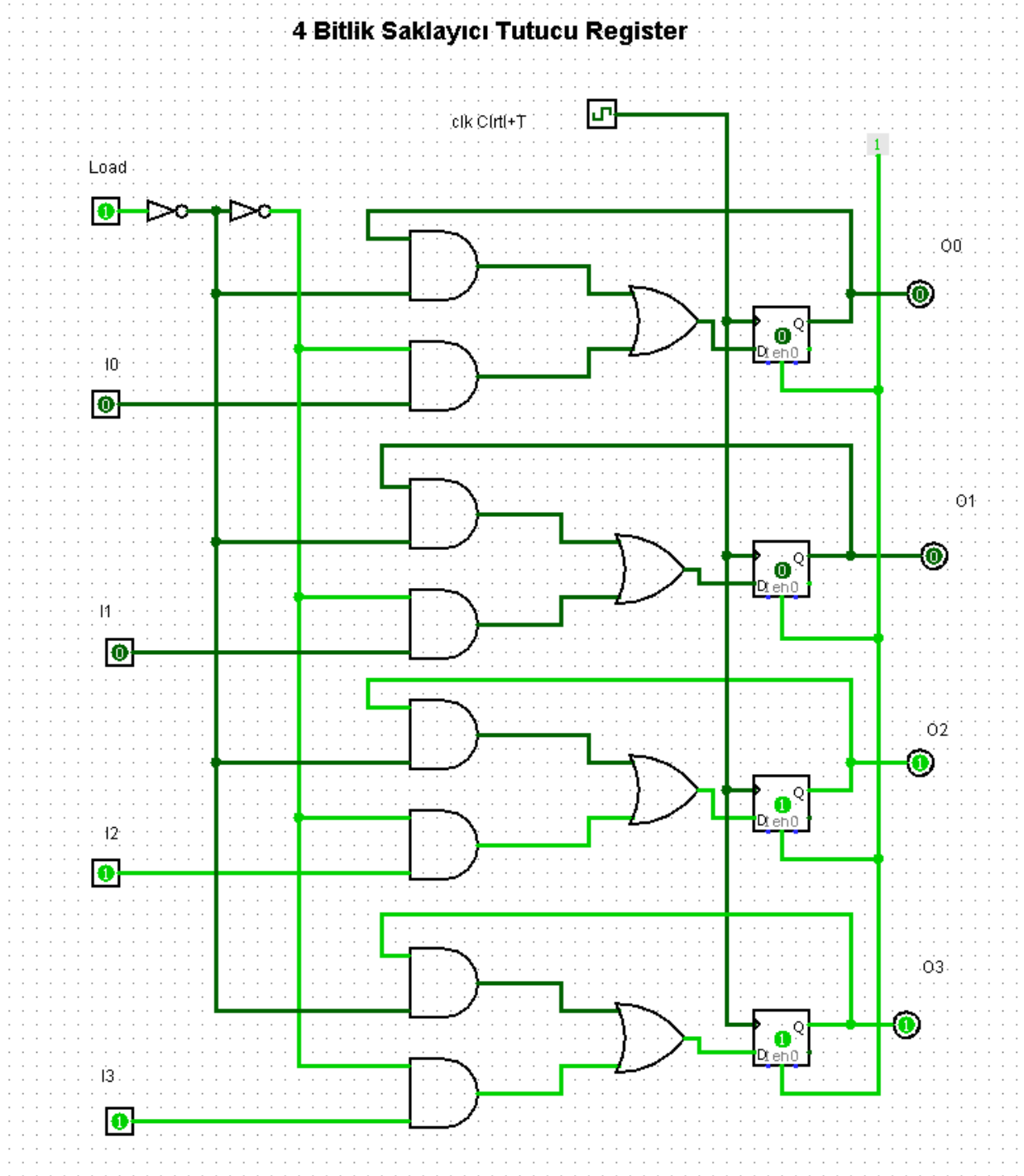


<b>TR</b>	16	Geçici Saklayıcı	Geçici verinin saklanabileceği saklayıcıdır.
<b>INPR</b>	8	Giriş Saklayıcı	8 bitlik karakterin, kesilim (interrupt) işlevi yoluyla, giriş ve çıkış bilgilerinin okunması ve yazılmasıyla görevlidir.
<b>OUTR</b>	8	Çıkış Saklayıcı	

Saklayıcılarda genel olarak üç kontrol işareti vardır: Bunlar **LD** (yükle), **INC** (arttır) ve **CLR** (sil) girişleridir. Bu tip bir saklayıcı yapısı için aşağıdakine benzer paralel yüklemeli, saat ve silmeli ikili sayıcıya ihtiyaç vardır.



Sadece yükle girişinin olduğu saklayıcılarda ise aşağıdaki gibi paralel yükleme saklayıcı devresi kullanılabilir



### 5.3 Bilgisayarın Bus Yapısı

Aşağıda örnek bir temel bilgisayarın yapısı gösterilmiştir. Temel bilgisayar sekiz saklayıcı, bir ana bellek ve kontrol biriminden oluşmaktadır. Bu yapıda ikili bilgiler, 16 bit hat grubu (Bus) yoluyla *Anabellek* ile saklayıcılar arasında ve saklayıcıların kendi aralarındaki veri aktarımı sağlar.

Ana belleği adres girişi sadece **AR** saklayıcısının çıkışına bağlıdır. Dolayısıyla Anabelleğe erişim mutlaka **AR** saklayıcısı üzerinden gerçekleştirilir. Bu şekilde her bir saklayıcının Anabelleğe erişimi için ayrı bir bus ihtiyacı giderilmiş olur. Benzer şekilde **AC** saklayıcısı Anabelleğe **AR** üzerinden erişebilir.

Buna karşın **ALU** ile **AC** arasındaki doğrudan bağlantıya dikkat edilmelidir. **ALU**' da yapılan tüm işlemlerde bir operand **AC** üzerinden işlem görür ve işlem sonucu ise yine **AC** 'ye aktarılır. Bu yapıya "tek akümülatörlü" yapı adı da verilir.

**ALU** nun diğer giriş operantı 3 farklı olabilir.

- i. Birinci durumda **ALU** nun tek girişi vardır ve bu giriş **AC** nin çıkışından gelir. **AC** nin değerini al veya **AC** nin içeriğini kaydır gibi komutlar bu şekilde gerçekleşir.
- ii. İkinci durumda girişler **DR** saklayıcısı üzerinden gelir. **AR** ve **DR** girişlerinin birlikte kullanılması ile matematiksel ve mantıksal komutlar gerçekleştirilir. Örneğin **AC** ile **DR** yi topla , **AR** ile **DR** Ve işlemi uygula . Bu işlemlerde sonuç gene **AC** de ve **E** (elde) bayrağını tutan FF de saklanır.
- iii. Üçüncü durumda giriş **INPR** saklayıcısından gelir.

**ALU**' da **E** bayrağı (**flag**) toplama işleminde Elde, çıkarma işleminde Ödücü, taşma gibi değerleri göstermekte kullanılır. Bu yapıda saklayıcılar arası bilgi aktarımı saklayıcıların çıkışları **S2 S1 S0** seçim girişleri ile kontrol edilen **MUX (3X8)** tarafından kontrol edilerek Bus üzerinden gerçekleştirilir.

**INPR** ve **OUTR** ise 8 bit içerik olduğundan, Bus içeriğinin en az anlamlı 8 bitini (**LSB**) kullanırlar.

**Örnek-1:** **S2 S1 S0 = 011** **DR** nin seçilmesini sağlar. Böylece, **DR** nin içeriği Bus' a verilir. Bu içerik hangi saklayıcıya yüklenecekse, onun "**yükle-LD**" kontrolü aktive edilir.

**Örnek-2:** **AC** nin "**yükle -LD**" hattı aktive edilirse **AC <= DR** transferi sağlanır, 16 bitlik bilgi **DR** den **AC** ye yazılır.

**Örnek-3:**

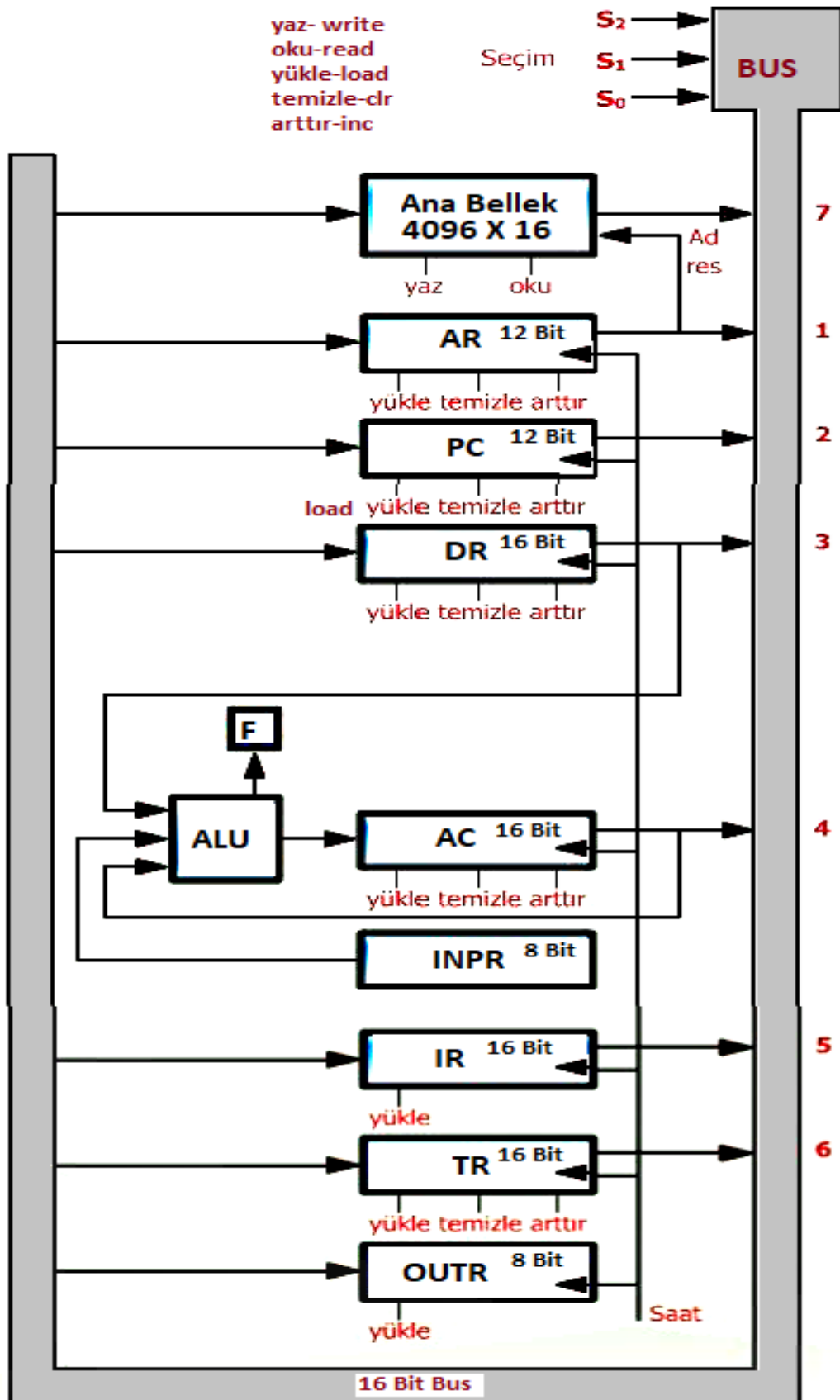
Bu yapıda herhangi bir saklayıcının içeriğinin Bus üzerine aktarılması ve Matematiksel, mantıksal işlemler aynı saat dilimi içinde gerçekleştirilebilir.

**DR <= AC, AC <= DR,**

Bu işlemde, **AC** içeriğini Bus üzerine yerleştir (S2 S1 S0 = 100) ve **DR** 'nin **LD** (yükle) girişini etkinleştirerek, **DR** içeriğini **ALU** üzerinden **AC** 'ye **AC** nin **LD** (yükle) ' girişini etkinleştirerek yapılabilir.

**Örnek-4: Anabellekten Okuma işleminde;** **AR** Anabellek adresini tutar, Anabelleğin Oku girişi (**Read**) seçilir ve **DR** saklayıcısının **LD** kontrolü aktive edilerek içerik **DR** ye transfer gerçekleşir.

**Örnek-5: Anabelleğe Yazma işleminde;** **AR** Anabellek adresini tutar, Anabelleğin Yaz (**Write**) girişi seçilir ve **DR** nin içeriği Anabelleğe transfer edilmiş olur.



## 5.4 Komut Yapısı

Temel bilgisayar için üç komut kodu formatı aşağıda gösterilmiştir. Her format 16 bit olup **opcode** kısmı 3 bit' tir. Bellek adreslemeli bir komutta 1 bit (MSB) adresleme modu için ve geri kalan 12 bit adres için kullanılır. Saklayıcı Adreslemeli komutlara 0111 ve giriş/çıkış komutlarına 1111 opcode değeri atanmıştır.



### Bellek Adreslemeli Komut Listesi

Sembol	Doğrudan Adresleme (I=0)		Dolaylı Adresleme (I=1)		Koşul	İşlem	Açıklama
	D <sub>7</sub> I' T <sub>3</sub>	Binary	Hex	Binary	Hex		
AND	0000. xxxx. xxxx. xxxx	0xxx	1000. xxxx. xxxx. xxxx	8xxx	D <sub>0</sub>	AC ← AC ∧ M[AR]	Anabellekteki kelimeyi <b>VE</b> leyrek AC ye aktar
ADD	0001. xxxx. xxxx. xxxx	1xxx	1001. xxxx. xxxx. xxxx	9xxx	D <sub>1</sub>	AC ← AC + M[AR], E ← Co	Anabellekteki kelimeyi <b>Toplayarak</b> AC ye aktar
LDA	0010. xxxx. xxxx. xxxx	2xxx	1010. xxxx. xxxx. xxxx	Axxx	D <sub>2</sub>	AC ← M[AR]	Anabellekteki kelimeyi <b>AC</b> ye yükle
STA	0011. xxxx. xxxx. xxxx	3xxx	1011. xxxx. xxxx. xxxx	Bxxx	D <sub>3</sub>	M[AR] ← AC	<b>AC</b> nin içeriğini Anabelleğe sakla
BUN	0100. xxxx. xxxx. xxxx	4xxx	1100. xxxx. xxxx. xxxx	Cxxx	D <sub>4</sub>	PC ← AR	Şartsız dallan
BSA	0101. xxxx. xxxx. xxxx	5xxx	1101. xxxx. xxxx. xxxx	Dxxx	D <sub>5</sub>	M[AR] ← PC, PC ← AR+1	Dallan ve Geri dönüş adresini sakla
ISZ	0110. xxxx. xxxx. xxxx	6xxx	1110. xxxx. xxxx. xxxx	Exxx	D <sub>6</sub>	M[AR] ← M[AR]+1, Eğer (M[AR]+1=0) ise PC ← PC+1	Arttır ve eğer sıfır ise atla

#### 5.4.1 Saklayıcı Adreslemeli Komut Listesi

(r=D<sub>7</sub> I' T<sub>3</sub> ; Bu koşul bütün Saklayıcı Komutlarındada ortaktır.)

Sembol	Binary	Hex	Koşul	İşlem	Açıklama
			r	SC ← 0	
CLA	0111. 1000. 0000. 0000	7800	rB <sub>11</sub>	AC ← 0	AC yi sil
CLE	0111. 0100. 0000. 0000	7400	rB <sub>10</sub>	E ← 0	E yi sil
CMA	0111. 0010. 0000. 0000	7200	rB <sub>9</sub>	AC ← AC'	AC nin tümleyenini al
CME	0111. 0001. 0000. 0000	7100	rB <sub>8</sub>	E ← E'	E nin tümleyenini al

<b>CIR</b>	0111. 0000. 1000. 0000	7080	<b>rB<sub>7</sub></b>	$AC \leftarrow shr \ AC, \ AC(15) \leftarrow E, \ E \leftarrow AC(0)$	<b>AC</b> ve <b>E</b> nin içeriğini sağa dairesel kaydır
<b>CIL</b>	0111. 1000. 0100. 0000	7040	<b>rB<sub>6</sub></b>	$AC \leftarrow shl \ AC, \ AC(15) \leftarrow E, \ E \leftarrow AC(15)$	<b>AC</b> ve <b>E</b> nin içeriğini sola dairesel kaydır
<b>INC</b>	0111. 1000. 0010. 0000	7020	<b>rB<sub>5</sub></b>	$AC \leftarrow AC+1$	<b>AC</b> nin içeriğini 1 arttır
<b>SPA</b>	0111. 1000. 0001. 0000	7010	<b>rB<sub>4</sub></b>	Eğer $(AC(15)=0)$ ise $(PC \leftarrow PC+1)$	<b>AC</b> pozitif ise bir sonraki komuta atla
<b>SNA</b>	0111. 1000. 0000. 1000	7008	<b>rB<sub>3</sub></b>	Eğer $(AC(15)=1)$ ise $(PC \leftarrow PC+1)$	<b>AC</b> negatif ise bir sonraki komuta atla
<b>SZA</b>	0111. 1000. 0000. 0100	7004	<b>rB<sub>2</sub></b>	Eğer $(AC=0)$ ise $(PC \leftarrow PC+1)$	<b>AC</b> sıfır ise bir sonraki komuta atla
<b>SZE</b>	0111. 1000. 0000. 0010	7002	<b>rB<sub>1</sub></b>	Eğer $(E=1)$ ise $(PC \leftarrow PC+1)$	<b>E</b> sıfır ise bir sonraki komuta atla
<b>HLT</b>	0111. 1000. 0000. 0001	7001	<b>rB<sub>0</sub></b>	$S \leftarrow 0$ (S başlangıç FF sidir.)	Programı durdur

### 5.4.2 Giriş/Çıkış Adreslemeli Komut Listesi

(**p=D<sub>7</sub> I T<sub>3</sub>** ; Bu koşul bütün Giriş /Çıkış Komutlarında ortakdır.)

Sembol	Binary	Hex	Koşul	İşlem	Açıklama
			<b>p</b>	$SC \leftarrow 0$	
<b>INP</b>	1111. 1000. 0000. 0000	F800	<b>pB<sub>11</sub></b>	$AC(0-7) \leftarrow INPR, FGI \leftarrow 0$	Giriş karakterini <b>AC</b> ye al
<b>OUT</b>	1111. 0100. 0000. 0000	F400	<b>pB<sub>10</sub></b>	$OUTR \leftarrow AC(0-7), fG0 \leftarrow 0$	<b>AC</b> nin içeriğini çıkışa ver
<b>SKI</b>	1111. 0010. 0000. 0000	F200	<b>pB<sub>9</sub></b>	Eğer $(FGI=1)$ ise $(PC \leftarrow PC+1)$	Giriş bayrağını atla
<b>SKO</b>	1111. 0001. 0000. 0000	F100	<b>pB<sub>8</sub></b>	Eğer $(FGO=1)$ ise $(PC \leftarrow PC+1)$	Çıkış bayrağını atla
<b>ION</b>	1111. 0000. 1000. 0000	F080	<b>pB<sub>7</sub></b>	$IEN \leftarrow 1$	<b>Kesme</b> yi aktif yap.
<b>IOF</b>	1111. 0000. 0100. 0000	F040	<b>pB<sub>6</sub></b>	$IEN \leftarrow 0$	<b>Kesme</b> yi pasif yap.

### 5.5 Tam Komut Seti ve Verimliliği

Genel olarak bir komut seti aşağıda tanımlanan her bir sınıftan yeteri komut bulunduruyorsa, bu set “tam bir komut seti” dir denilir. Bu sınıflar:

- Aritmetik, Lojik ve Kaydırma komutları
- Anabellek ile saklayıcılar arasında bilgi transfer komutları
- Program kontrol komutları
- Giriş / Çıkış komutları

Temel bilgisayar komutları tablosunda tanımlanan komutların her bir sınıftan “minimum” komut içerdiğini göstermek mümkündür. Böylece tanımlanan set, bir minimum settir.

Örneğin, sadece ADD operasyonu vardır, çıkarma için 2-komplement elde edilip, toplama yapılması gerekir. Çarpma, bölme vs. komutları için aynı yol izlenmelidir.

Diğer taraftan, bu komut seti verimli bir set değildir. Sık kullanılan komutlar hızlı bir şekilde gerçekleşmemiştir. Çıkarma, çarpma işlemleri çok yavaş gerçekleşmiştir. Günümüz bilgisayarları hızlı devrelerle bu işlemleri yapabilmektedir.

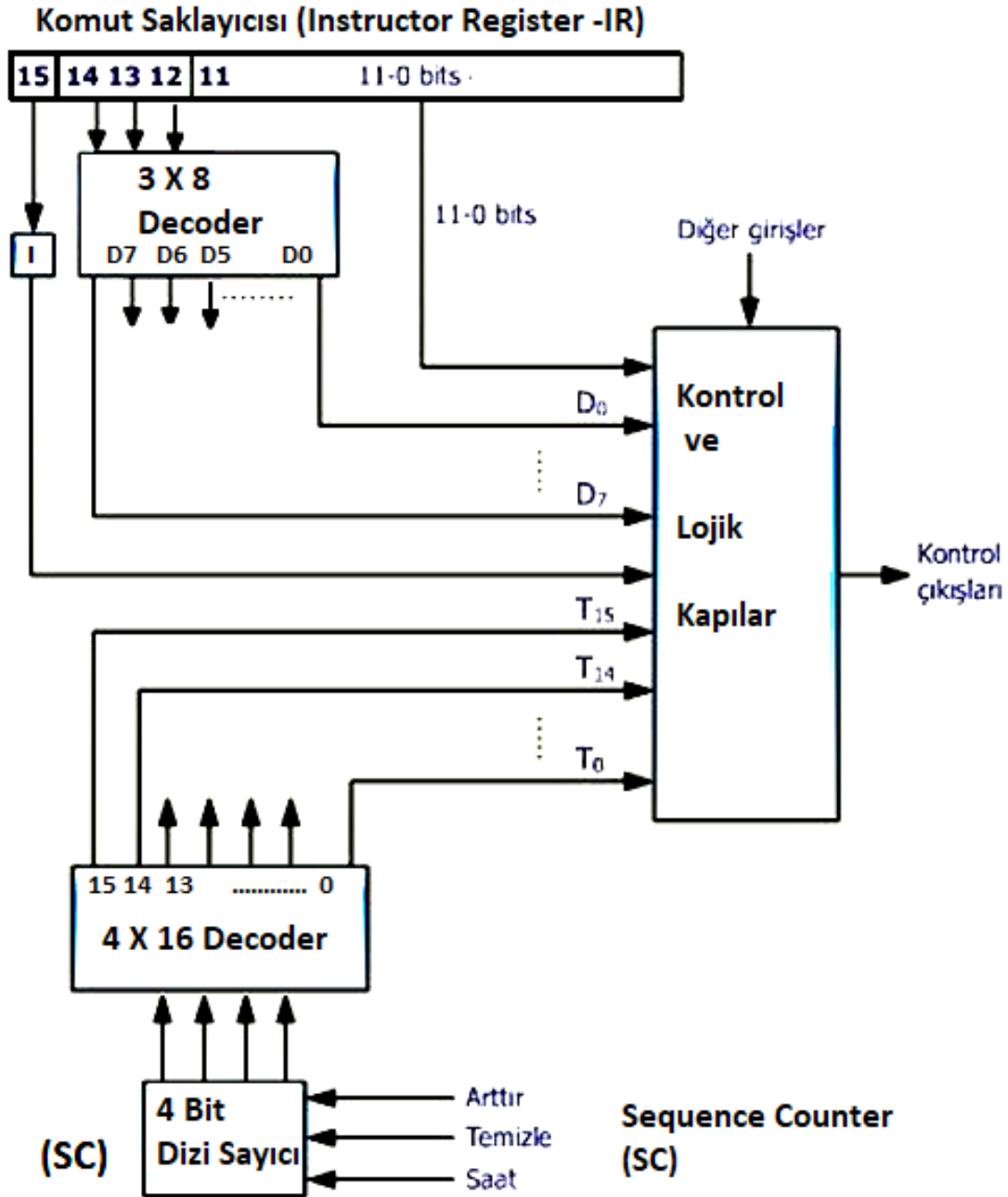
## 5.6 Zamanlama ve Kontrol

Bilgisayarın zamanlaması master saat üretici tarafından oluşturulan saat darbeleriyle sağlanır. Bu devre tipik bir ikili sayıcı ve dekode devresi olarak tasarlanabilir. Saat darbeleri ancak kontrol işareti ile birlikte olunca saklayıcıların durumunu değiştirebilir.

İki çeşit kontrol yapısı kullanılmaktadır: donanım yoluyla (hardwired) ve mikroprogramlama veya yazılım yoluyla (microprogrammed) kontrol. Donanım yoluyla kontrolde, lojik kapılar, FF' ler, dekode ve diğer sayısal devreler kullanılır. Bu kontrol hızlıdır, bir kez tasarlanınca bir daha değiştirilemez. Mikroprogramlama yoluyla kontrol esnektir fakat daha yavaştır. **Bu bölümde, donanım yoluyla kontrolden bahsedeceğiz.**

Aşağıda donanım yoluyla kontrol blok diyagramı gösterilmiştir.





### 5.6.1 Kontrol Birimi

Hafızadan okunan komut **IR** saklayıcısında tutulur. Bu saklayıcının 3 parçası vardır.

- **I** biti; MSB biti olup doğrudan (**I=0**) ve dolaylı (**I=1**) adresleme modunu seçmeçtedir.
- İşlem kodu (**opcode**); 12, 13 ve 14. Bitler olup komutun opconunu göstermektedir.
- Adres; 0-11 .bitler olup adresleme moduna göre komutun adresini veya adresini tutan saklayıcıyı göstermektedirler.

İşlem kodunun 12, 13 ve 14. Değerleri 3X8 lik kod çözücüler ile çözülerek kod çözücünün çıkışındaki D0,D1, .. D7 çıkışlarını oluşturur. **I** biti ile beraber zamanla ve kontrol ünitesinin kontrol kısmını gösterirler.

### 5.6.2 Zamanlama Birimi

Zamanlama kısmında bulunan 4-bit dizi sayıcısı (sequence counter, **SC**) 0000'dan 1111'e kadar sıra ile sayarak, önündeki 4x16'lık dekode devresinin 0'dan 15'e kadar çıkışlarından birini seçmektedir. Seçilen çıkış T0'dan T15'e kadar bir darbe üretir.

Zamanlama ve Kontrol devresi çıkışı, sonuçta komut ile ilişkili bir kontrol işareti dizisi oluşturur. Bunu bir örnek ile ve zamanlama diyagramı ile açıklayalım:

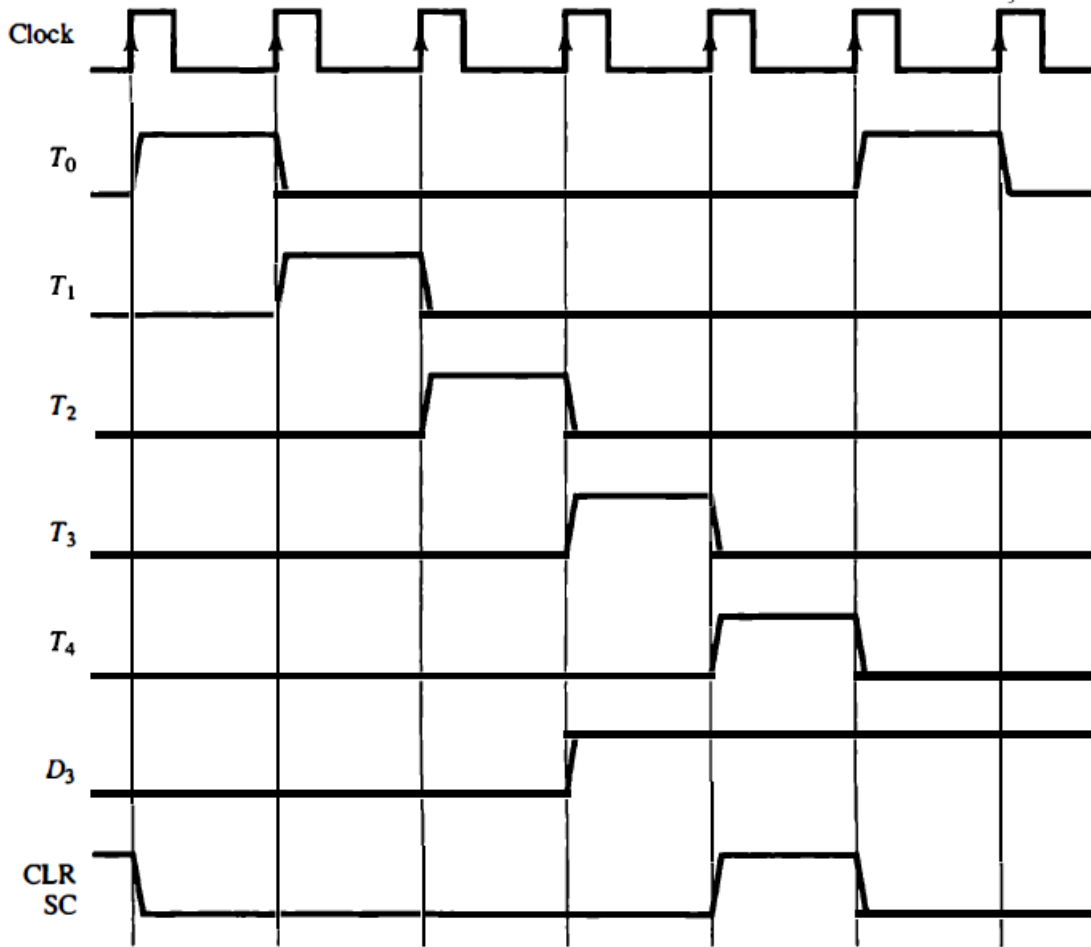
#### Örnek:

**D<sub>3</sub> T<sub>4</sub> : SC ≤ 0**

Yukardaki ifadenin zamanlama diyagramı aşağıda gösterilmiştir. İfadenin koşulu (D<sub>3</sub>T<sub>4</sub>) sağlandığında, dizi sayıcısının (SC) saat darbesinin pozitif geçişinde “temizle” girişi aktif yapılarak ifade (**SC = 0**) sağlanır.

Saat darbe üretici, dekode çıkışından T<sub>0</sub> ,T<sub>1</sub>, T<sub>2</sub>, T<sub>3</sub>, T<sub>4</sub> darbelerini üretir. D<sub>3</sub> komut dekode çıkışı ve T<sub>4</sub> işareti ikisi de aktif olduklarında üretilen çıkış, SC'nin “ temizle” girişine uygulanarak saat darbe üreticinin tekrar T<sub>0</sub> 'dan başlaması sağlanır.

**D<sub>3</sub> T<sub>4</sub> : SC ≤ 0**



### Örnek:

Örnek olarak aşağıdaki saklayıcı transfer işlemini örnek alalım ;

**T<sub>0</sub> : AR ≤ PC**

Bu işlem eğer zamanlama sinyali T<sub>0</sub> aktifse, **PC** içeriğinin **AR**'ye bir transferini belirtir. T<sub>0</sub> saat döngüsü aralığının tamamı boyunca aktiftir. Bu süre zarfında **PC** içeriği Bus üzerine (S<sub>2</sub>S<sub>1</sub>S<sub>0</sub> = 010) seçilerek yüklenir. **PC**'in içeriği **AR**'nin **LD** girişi etkinleştirilerek yerleştirilir. Asıl aktarma, saat döngüsünün sonuna kadar gerçekleşmez. Bu işlem sonunda aynı zamanda **SC** sıra sayacını 0000 olur.

## 5.7 Komut Fazları

Ana bellekte saklanmış olan programın her bir komutu, aşağıdaki fazlarla koşturulur:

- Ana bellekten bir komutun alınması (fetch cycle)
- Komutun dekode edilmesi (decode cycle)
- Efektif adresin hesaplanması ve komutun dolaylı adresinin olup olmadığının araştırılması
- Komutun koşturulması (execution cycle)

Komutun koşturulmasından sonra bir sonraki komut alıp getirelerek HALT komutuna kadar işlem devam ettirilir.

### 5.7.1 Komutun alınması

Komutun alınması ve dekode edilmesi (fetch and decode cycle), aşağıdaki bellek transfer dili ifadeleri ile tanımlanır:

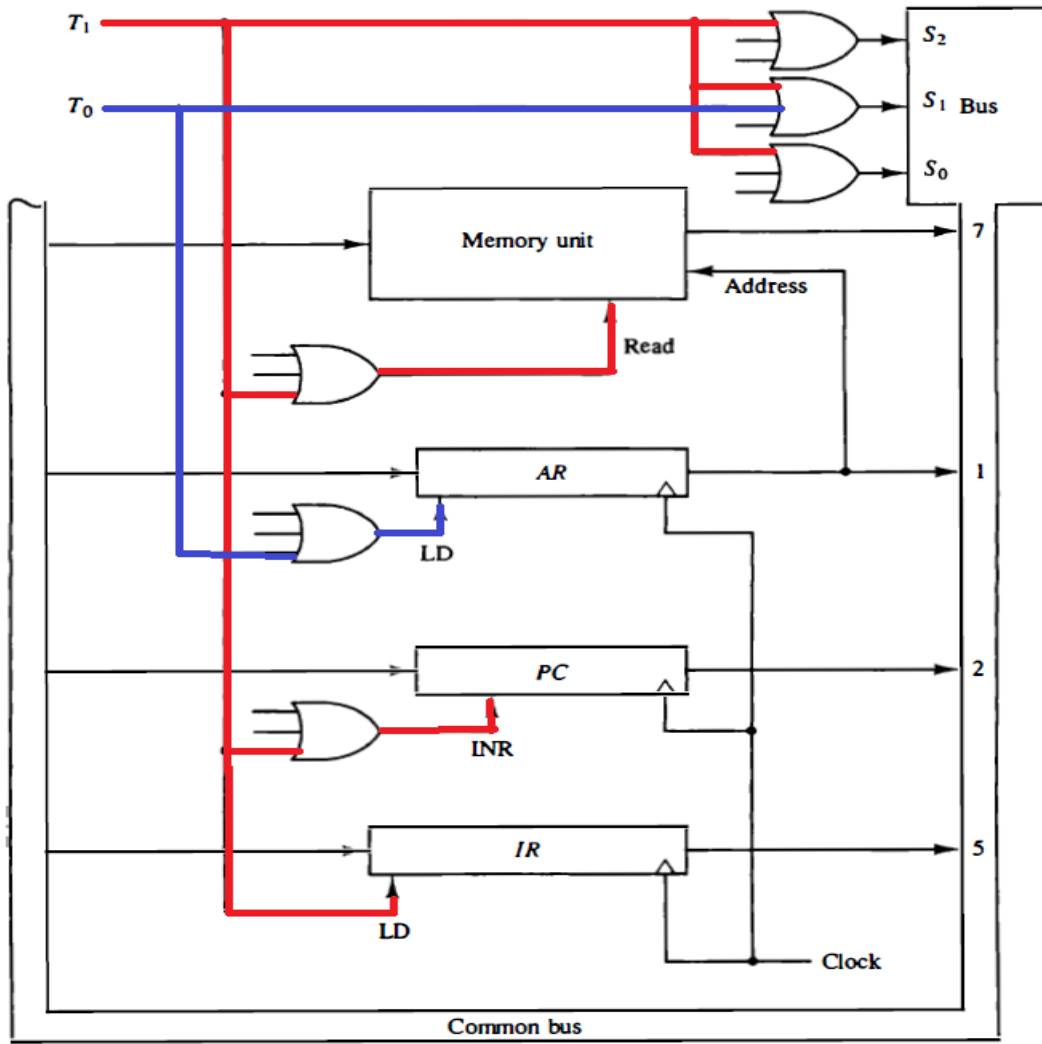
<b>T0 : <math>AR \leq PC</math></b>
<b>T1 : <math>IR \leq M[AR], PC \leq PC+1</math></b>
<b>T2 : <math>D0, ..., D7 \leq \text{Decode } IR(12-14), AR \leq IR(0-11), I \leq IR(15)</math></b>

- i. Sadece **AR** saklayıcısını çıkışı Anabelleğin adres girişine bağlı olduğundan T0 saat geçişinde **PC**'de saklı olan komut adresi **AR**'ye transfer edilir.
- ii. T1 saat geçişinde iki işlem gerçekleşir: ilkinde, **AR** ile gösterilen Ana bellek adresteki içerik **IR**'ye aktarılır ikincide **PC**, 'in içeriği 1 arttırılır.

T1 saat geçişinde meydana gelen (  $IR \leq M[AR], PC \leq PC+1$  ) ifadesi aşağıdaki adımlarla gerçekleşir:

- a- Ana belleğin **Read** (Okuma) girişi aktive edilir.
- b- Bus selec girişine  $S_2S_1S_0=111$  verilerek Anabellek aktive edilir.
- c- **IR** saklayıcısının **LD** girişi aktive edilerek Anabelleğin çıkışı **IR** saklayıcısına yazılır.
- d- **PC** saklayıcısının **INC** kontrolü ile değeri 1 arttırılır.

Bu iki adım aşağıdaki şekilde verilmiştir.



- iii. T2 zamanında **IR**' de bulunan işlem kodu (**opcode**) çözülür, dolaylı biti **I FF** (flip-flop) 'sine aktarılır ve komutun adres kısmı **AR**' ye taransfer edilir.

## 5.8 Komut Yapıları

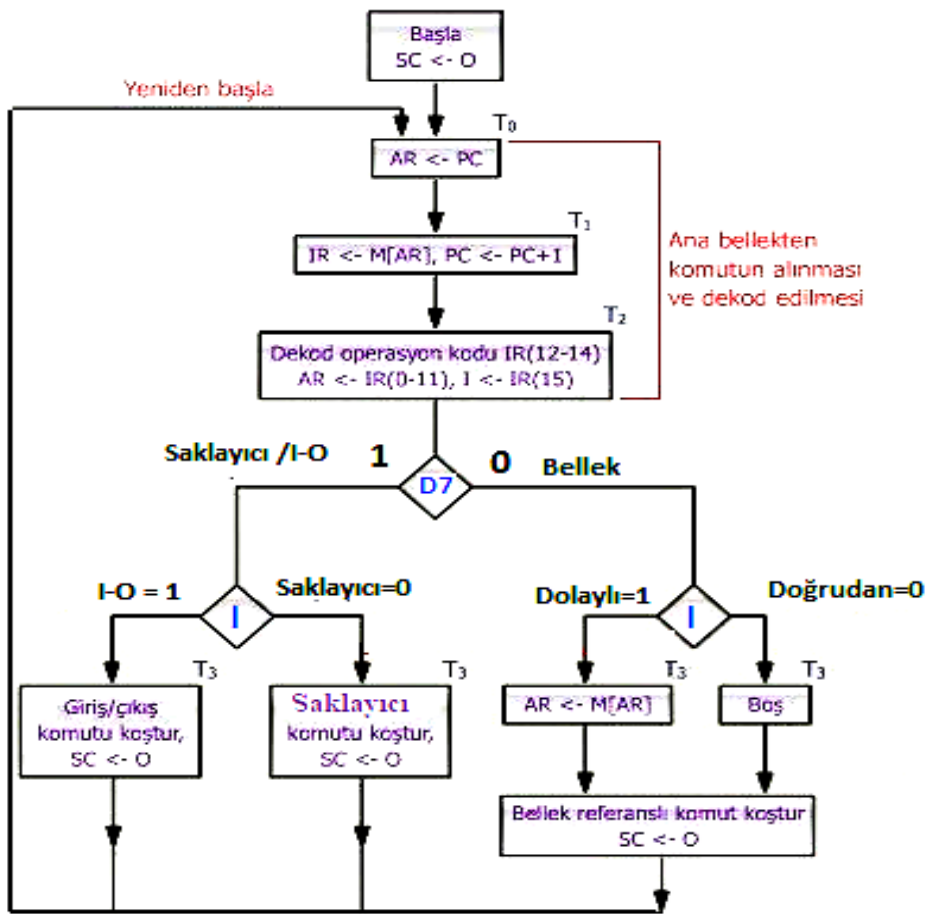
### 5.8.1 Komut Tipinin Belirlenmesi ve Dallanma

T3 zamanında ise ana bellekten okunan komutun tipinin belirlenmesi fazı gelir. Üç komut tipine göre dört kola ayırımın gerekir;

D7 I T3 : Giriş/çıkış komutu
D7 I' T3 : Saklayıcı Adreslemeli komut
D'7 I T3 :Dolaylı Bellek Adreslemeli komut
D'7 I' T3 : Doğrudan Bellek adreslemeli komut (İşlem yok)

Herbir komut yürütüldükten sonra, SC sıfırlanır. SC saat işaretinin her yükselen kenarında sıfırlanır veya artırılır. SC nin arttırımı için  $SC \leftarrow SC + 1$  ifadesi yazılmayacaktır, ancak kontrolün sırayla bir sonraki zamanlama sinyaline gittiğini ima eder. Ne zaman SC sıfırlanacaksa,  $SC \leftarrow 0$  ifadesini ekleyeceğiz.

Komutun yukarda anlatılan fazları ile koşturulması, “saklayıcı transfer dili” yanında “akış diyagramı” ile de gösterilebilir. Aşağıda komut fazlarının koşturulumu akış diyagramı ile gösterilmiştir.



### 5.8.2 Saklayıcı Adreslemeli Komutlar

Temel bilgisayarda, Saklayıcı Adreslemeli komutlar  $D7 = 1$  ve  $I = 0$  kontrol değerleri ile tasarlanmışlardır. Komut kodunun 0' dan 11' e kadar olan değerleri ile tanımlanan 12 komuttan oluşurlar. İlk yedi tanesi AC ve E bayrağı üzerinde tanımlanmış temizleme, değili alma (komplement), dairesel kaydırma ve arttırma işlemlerinden oluşurlar( CLA, CLE, CMA, CME, CIR, CIL, INC)

Kalan dört tanesi komuttan sonrakine atlama ve sonuncusu ise işlemi durdurma olarak tanımlanmıştır. SPA ve SNA işaretli sayılarla yapılan işlemlerde, SZA sonucun sıfır olup olmadığının, SZE ise sonuçta elde olup olmadığının kontrol işleminde kullanılır.

Bilindiği gibi

**$D_7 I' T_3 : d$  ve  $IR(i) = B_i$  ( $IR(011)$  bit değerleri)**

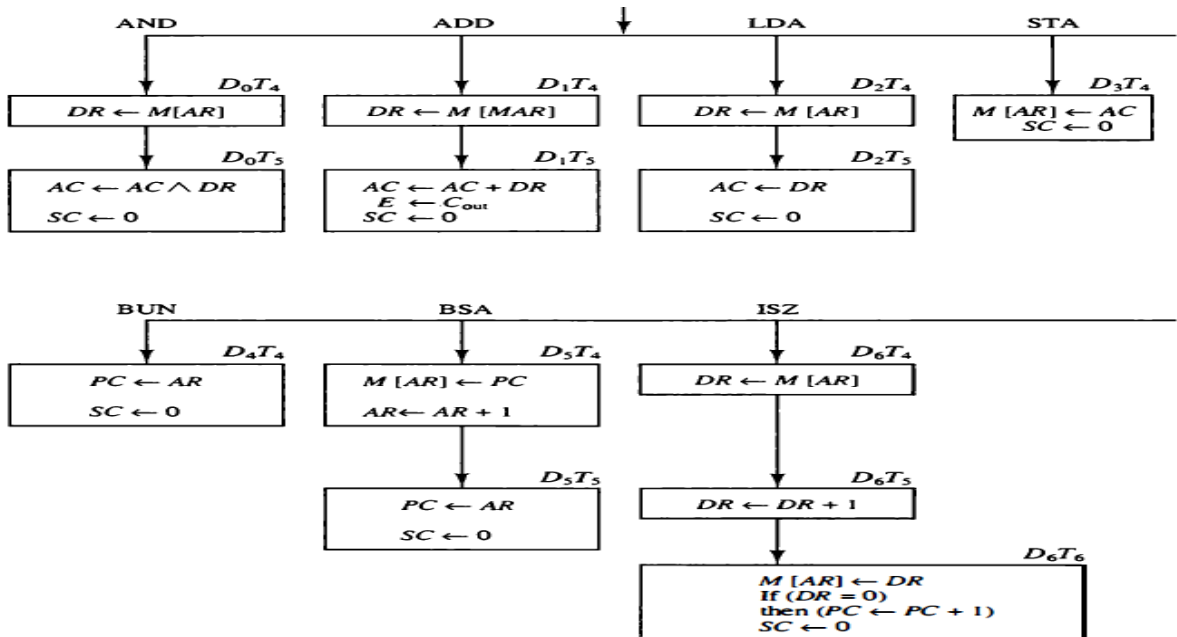
ile tanımlanırlar. Bütün Saklayıcı Adreslemeli komutlarda ( $r = D_7 I' T_3$ ) ortaktır ve Saklayıcı adreslemeli veya Giriş-Çıkış adreslemeli bir komut  $T_3$  periodunda çalıştırılabilir.

### 5.8.3 Ana Bellek Adreslemeli Komutlar

Yedi ana bellek referanslı komut,  $D_i$   $i=0, 1, \dots, 6$  operasyon koduna göre AND, ADD, LDA, STA, BUN, BSA ve ISZ olarak sıralanır.

$I=0$  olan bir bellek adreslemeli komutu ile karşılaşıldığında (doğrudan adresleme), etkili adres  $T_2$  periodunda AR saklayıcısına yüklendiği için  $T_3$  peridunda bir şey yapmak gerekmez. Ancak, bellek adreslemeli bütün komutlar  $T_4$  periodunda gerçekleştiği için SC bir arttırılmalıdır.

$I=1$  olan bir komut işlendiğinde (dolaylı adresleme) ise  $T_3$  periodunda adres Anabellekten okunup AR saklayıcısına yüklenir.



#### 5.8.3.1 AND: AC nin mantıksal ve işlemi.

Bu işlem aşağıdaki mikroişlemlerle gösterilir:

D0 T4 : $DR \leq M[AR]$
-------------------------

D0 T5 : $AC \leq AC \text{ AND } DR, SC \leq 0$
---

D0 T5 fazında AND işlemi tamamlanıp, aynı saat darbesinde  $SC \leq 0$  mikroişlemi ile yeni komutun mikroişlem dizisine başlanır.

#### 5.8.3.2 **ADD:** AC nin içeriğini Topla

Bu işlem aşağıdaki mikroişlemlerle gösterilir

D0 T4 : $DR \leq M[AR]$
-------------------------

D0 T5 : $AC \leq AC + DR, E \leq C_{out}, SC \leq 0$
--

olarak tanımlanır. Cout ise toplama mikroişleminin eldesidir.

#### 5.8.3.3 **LDA :** (Load Accumulator) AC' ye Yükle

Bu işlem aşağıdaki mikroişlemlerle gösterilir

D2 T4 : $DR \leq M[AR]$
-------------------------

D2 T5 : $AC \leq DR, SC \leq 0$
---------------------------------

AC saklayıcısı ile ortak BUS arasında direkt bir giriş bağlantısı olmadığı için, Anabellek üzerindeki data DR saklayıcısı üzerinden yüklenir.

#### 5.8.3.4 **STA** (Store Accumulator) AC içeriğini Sakla.

D3 T4 : $M[AR] \leq AC, SC \leq 0$ dır.
---

Burada AC saklayıcısı ile ortak BUS arasında direkt bir çıkış bağlantısı olduğuna dikkat edilmelidir.

#### 5.8.3.5 **BUN:** (Branch Unconditionally) Koşulsuz Dallanma

D4 T4 : $PC \leq AR, SC \leq 0$
---------------------------------

olup, efektif adres PC'ye aktarılır ve program akışı bu noktadan devam eder.

PC nin mikroişlemcinin işleyeceği bir sonraki komutun adresini tuttuğuna dikkat edilmelidir. Bu komut ile mikroişlemcinin adışıl bir sıra ile komutları çalıştırması engellenmiş olur.



### 5.8.3.6 **BSA:** (Branch Subroutine) Alt Programa Dallan

Bu komut bir alt programa (fonksiyon, prosedür) dallanma için kullanılır. BSA veya dallan ve geri dönüş adresini sakla komutunun mikroişlemleri aşağıdaki şekilde gösterilir

D5 T4 : $M[AR] \leq PC, AR \leq AR+1$
---------------------------------------

D5 T5 : $PC \leq AR, SC \leq 0$
---------------------------------

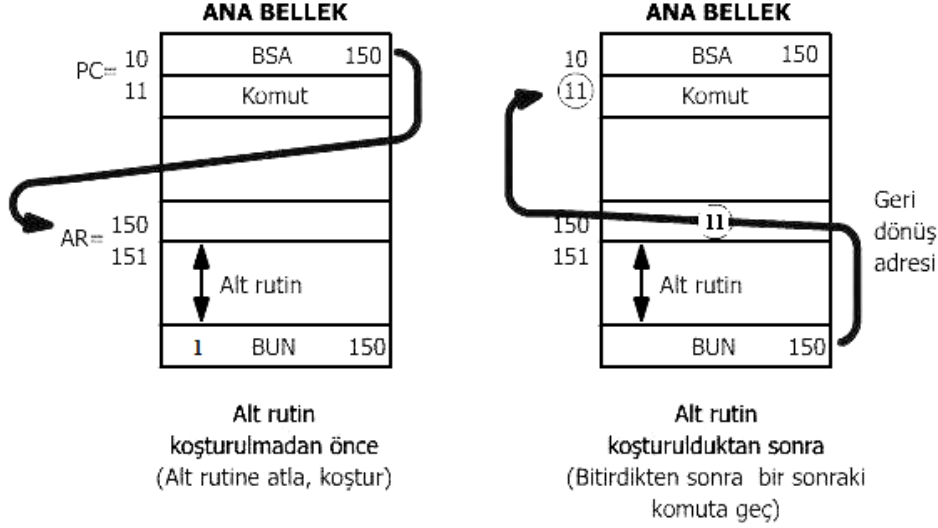
BSA komutu koşturulduğunda, T4 periodunda bir sonraki komutun adresini operant üzerinde belirtilen adres bölgesine depolar (Burada bir sonraki komutun adresinin PC de olduğuna dikkat edilmelidir) ve operant üzerindeki belirtilen adres değeri bir atırılarak T5 periodunda PC ye aktarılır. ( $PC \leq AR+1$  ; Bu adres değeri alt programın başlangıç adres değeridir)

#### **BSA Koşturulmasına Ait Örnek:**

Aşağıdaki şekilde BSA koşturulması örnekle açıklanmıştır. Genellikle bir alt program işlevini yaptıktan sonra ana programa geri döner. Bu örnekte alt programa dallanma işlemi ve geri dönüş işlemi 2 ayrı şekilde verilmiştir.

Burada komut; **BSA 150** olarak verilmiştir. Alt program 151 adresinden itibaren başlamış olup, geri dönüş adres değeri olan 11 ise 150. adres bölgesinde saklanmıştır.

Alt program bittiğinde, alt programın son satırında dolaylı adresleme modunda **BUN [150]** komutu ile alt programın başlangıç adresinde tutulan geri dönüş adresi PC ye aktarılarak alt programa dallanma işlemi yapılmadan önceki adres bölgesine gelinmiş olur.



### 5.8.3.7 **ISZ:** (Increment and Skip if Zero) Arttır ve Sıfır ise atla;

Bu komut aşağıdaki şekilde tanımlanır;

D6 T4 : DR ≤ M[AR]

D6 T5 : DR ≤ DR+1

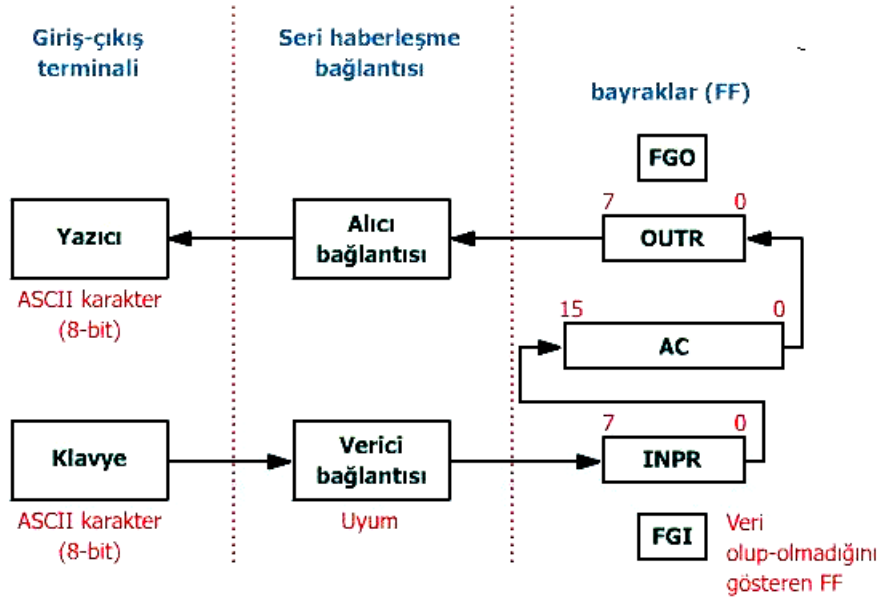
D6 T6 : M[AR] ≤ DR, Eğer (DR=0) ise (PC ≤ PC+1), SC ≤ 0

## 5.9 Giriş/Çıkış Ve Kesinti Kavramı

Bilgisayar dış dünya ile bağlantıyı giriş/çıkış düzenleri ile sağlar. Pek çok giriş/çıkış düzeni arasında en yaygın olanları klavye, tarayıcı, tablet (giriş için) ve yazıcı, ekrandır (çıkış için).

Örnek giriş/çıkış düzenimiz, klavye ve ekran ile temel bilgisayarın bağlantısını “kesinti” kavramı ile işletmektir. Bunun için giriş düzeni klavyeden basılan her tuşa karşı düşen ASCII karakter seri iletişim bağlantısı yoluyla INPR saklayıcısına gelir. Seri çıkış bilgisi ise OUTR saklayıcısında saklanır. Bu iki saklayıcı ACC saklayıcısı ile bağlantılıdır. Giriş –Çıkış işlemlerinde bilginin INPR geldiğini göstermek amacıyla bir 1 bitlik FGI (FlaG Input) bayrağı kullanılır. Örnek olarak klavyeden okunan bir tuşun ASCII kod karşılığı INPR saklayıcısına geldiğinde FGI bayrağı 1 olur.

Benzer şekilde yazıcıya göndermek istediğimiz bir byte’lık bir değer ACC saklayıcısından OUTR saklayıcısına geldiğinde FGO (FlaGOutput) bayrağı 1 olur. Bu şekilde mikroişlemci ile giriş-çıkış üniteleri arasında bağlantı sağlanmış olur.



Temel bilgisayarın giriş/çıkış komutları aşağıdaki tabloda gösterilmiştir. D7 I T3 : P bütün giriş/çıkış komutları için ortak koşuldur. Bu koşulda D7 =1, I=1, T3 =1 sonuçta da P=1 olması gerekir . Bütün Giriş-Çıkış komutları T3 periodunda gerçekleşir.

INP ve OUT komutları verinin INPR ve OUTR saklayıcısından AC saklayıcısına transferini gerçekleştirir. Burada AC saklayıcısının düşük değerli (0-7) bitleri kullanılır. INP ve OUT komutlarından sonra FGI ve FGO bayraklarının sıfırlandığına dikkat etmek gerekir.

SKI (SKip on Input flag) ve SKO (SKip on Output flag) komutları, FGI ve FGO bayraklarını kontrol edip, INPR ve OUTR saklayıcılarında veri olup, olmadığının anlaşılmasını sağlar.

(SKI, SKO) komutları bir atlama işlevi olup FGI ve FGO bayraklarını kontrol eder. SKI, SKO komutları BUN komutu ile birlikte bir döngü oluşturulur. Bayraklar (FGI, FGO) 0 ise dallanma komutu atlanmayacaktır. Bayraklar (FGI, FGO) 1 ise dallanma komutu (BUN) atlanacak ve giriş-çıkış komutları (INP, OUT) koşulacaktır.

### 5.9.1 Kesinti Programı ile Veri Giriş/Çıkışı

Kesinti (interrupt) programı ile veri giriş/çıkışının sağlanması için temel bilgisayarda gerekli yapı değişikliğinin sağlanması gerekir. Bununla ilgili temel bilgisayarın akış diyagramına bir kesinti fazı eklenmesi gerekir. Aşağıdaki şekilde kesinti fazı eklenmiş komut fazı görülmektedir.

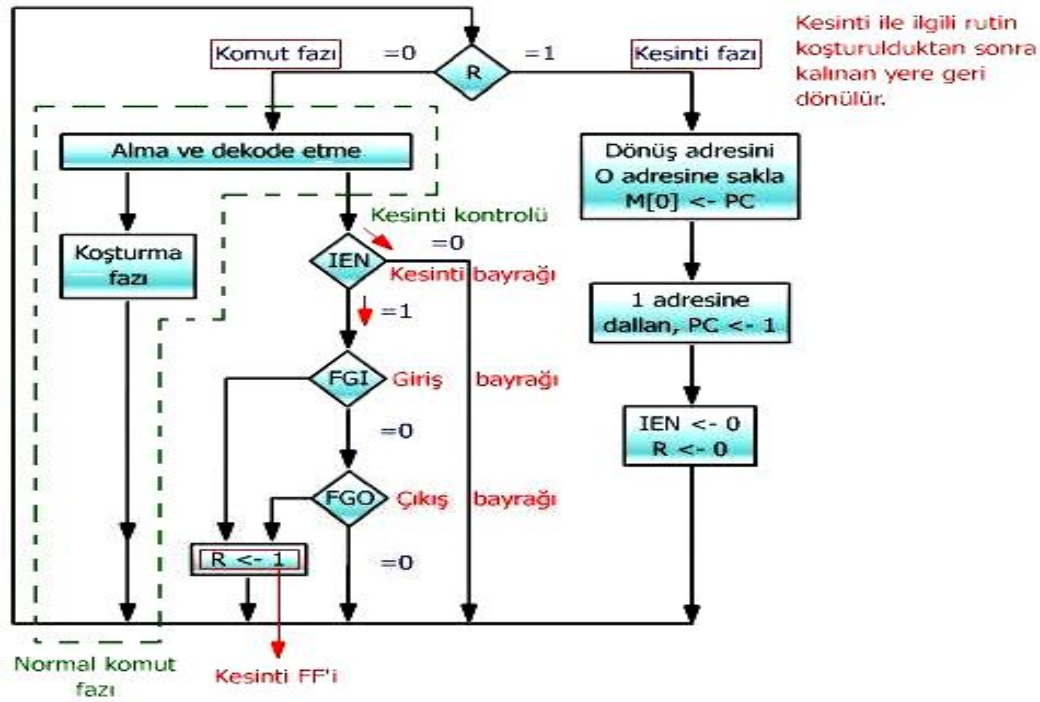
ION ve IOF kesilim-kesinti (interrupt) işleminin kontrolü için kesilim tetikleme bayrağını (IEN-Interrupt Enable Flag) kontrol etmek için kullanılır. IEN bayrağı 0 (IOF komutu ile) yapıldığında kesilim işlemi gerçekleşmez fakat IEN bayrağı 1 (ION komutu ile) yapıldığında kesilim işlemi gerçekleşir. Bu iki komut programcıya kesilimi kullanıp kullanmayacağına karar verme yeteneği sunar.

Kesilim işlevini gerçekleştirmek için mikroişlemciye R (kesilim) bayrağı ilave edilmiştir. R bayrağı 0 ise ( $R=0$ ) mikroişlemci kesilim fazına girmez ve komut fazını koşturur. Komut fazı esnasında IEN bayrağı kontrol edilir. IEN bayrağı 0 ise programcının kesilimi kullanmak istemediği anlaşılır ve komut fazına devam edilir. Eğer IEN bayrağı 1 ise FGI ve FGO bayrakları kontrol edilir. Eğer her iki bayrak 0 ise mikroişlemcinin bir giriş-çıkış işlemini yapmak için hazır olmadığını gösterir.

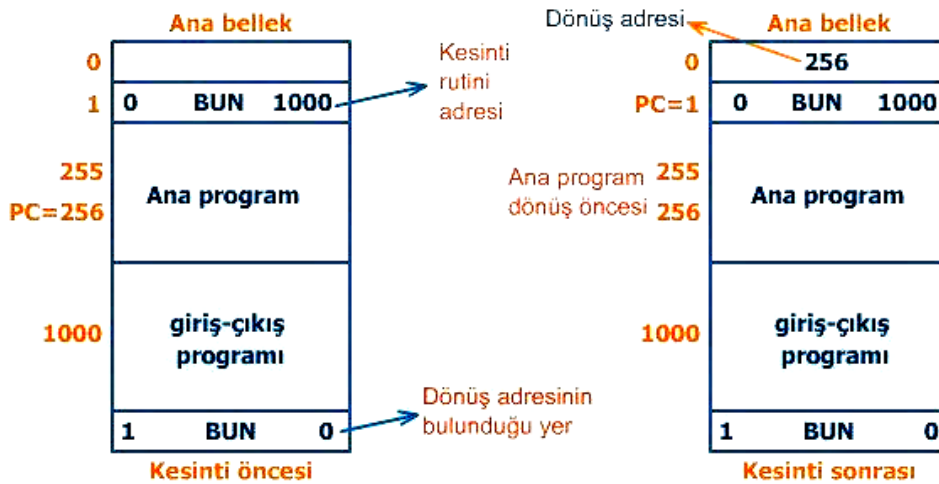
Kesilim fazı, kesilim bayrağı  $R=1$  yapıldıktan sonra yeni komut işleme periodunda başlar.  $R=1$  olması  $IEN=1$  veya FGI veya FGO 'nun 1 olmasına bağlıdır. Bu şekilde, kesilim fazı ile komut fazını değiştirmiş oluruz.

$T_0' T_1' T_2' (IEN)(FGI+FGO): R \leq 1$

.Kesilim fazında dallanma işlevi ve geri dönüş adresi donanımsal olarak tanımlanır. Bilgisayarlarda geri dönüş adresi saklayıcılar üzerinde, yığın üzerinde veya belirli bir adres bölgesinde olabilir. Temel bilgisayarda ise 0. Adres bölgesi geri dönüş adresini saklamak için kullanılmıştır.



Aşağıdaki şekilde kesilim ile bir giriş-çıkış işlevinin çalışması açıklanmıştır.



Bu örnekte başlangıçta kesilim programı (*interrupt handling routine*) 1000 adresinden itibaren hafızaya yüklenmiştir ve kesilim programının adresi 1 numaralı adres bölgesine yazılmıştır. (BUN 1000 ; doğrudan adresleme).

Kesilim programının son satırına geri dönüş işlemini yapabilmek için, kesilim olduktan sonra geri programın o andaki çalıştığı adresi gösteren yer olarak tanımlanan 0 adresi (*interrupt vector adres*) yazılmıştır (BUN [0] ; dolaylı adresleme).

Kesilim geldiğinde (R=1) 255 satırındaki komutun koşulduğunu düşünelim. Bu esnada PC de geri dönüş adresi olarak 256 değeri olacaktır. T0 periodunda (yeni komut başladığında) R=1 olduğundan kesilim fazi çalışmaya başlayacaktır. Bu esnadaki PC nin içeriği (256) anabelleğin 0

adres değerine, PC nin içeriğine ise kesilim programının adresini göstermek üzere 1 değeri yazılacaktır. İlave olarak kesilim bayrağı (R=0) ve kesilim tetikleme bayrağı (IEN=0) yapılacaktır.

R T0 : M[0] <= PC, PC<=1, R<=0, IEN=0

Mikroişlemci bir sonraki programı çalıştırdığında PC nin içeriğinde 1 olduğundan, 1 nolu adres bölgesindeki komut çalıştırılacaktır. Kesilim vektör adresinin içeriğinde bir koşulsuz dallanma komutu daha önceden yazılmış olduğundan (BUN 1000 ; doğrudan adresleme) program 1000 nolu satırdan itibaren koşturmayla başlayacaktır. 1000 adresinden itibaren koştur programın (kesilim servis programı/ giriş-çıkış programı) başında kesilimin hangi kaynaktan meydana geldiğini kontrol etmek için FGO ve FGI bayrakları kontrol edilir. Bu bayrakların durumuna göre gerekli komutlar çalıştırılır.

Bu kesilim servis programının son satırında geri dönüş adresini bulabilmek için (BUN [0] ; dolaylı adresleme) yazılır. Anabelleğin 0. adres bölgesinin içeriği PC ye yüklenerek program kaldığı yerden çalışmaya devam eder.

### 5.9.2 Güncellenmiş Komutun Alınması Fazı

Bu aşamada komut alma ve kod çözme fazlarını biraz değiştirelim. Komut çözme fazlarında sadece T0, T1 ve T2 periodlarını kullanma yerine R' (kesilim bayrağını) koşula (R'T0, R'T1 ve R'T2) dahil edelim. Bunun nedeni sadece R=0 olduğunda komut fazına girecektir. R=1 olduğunda ise kesilim fazı koşturulacaktır. Kesilim fazında PC üzerindeki geri dönüş adresi anabelleğin 0. adresine aktarılır, anabelleğin 1. adres bölgesine dallanılır, IEN ve R bayrakları sıfırlanır ve SC sıfırlanır. Bu işlemler aşağıdaki mikroişlemlerle verilmiştir.

RT0: AR<=0, TR<=PC

RT1: M[AR]<=TR, PC<=0

RT2: PC<=PC+1, IEN<=0, R<=0, SC<=0

İlk zaman işareti sırasında, AR temizlenir, PC içeriği TR geçici saklayıcıya aktarılır. İkinci zaman işareti sırasında, PC temizlenir, dönüş adresi bellek adresi 0'da saklanır. Üçüncü zaman işaretinde, PC artırılır, R ve IEN bayrakları 0 yapılır ve SC başa döner.

Yukarıdaki örnek için bu işlem aşağıdaki şekilde olur;

RT0: AR<=0, TR<=256

RT1: M[0]<=256, PC<=0

RT2:  $PC \leq 1$ ,  $IEN \leq 0$ ,  $R \leq 0$ ,  $SC \leq 0$

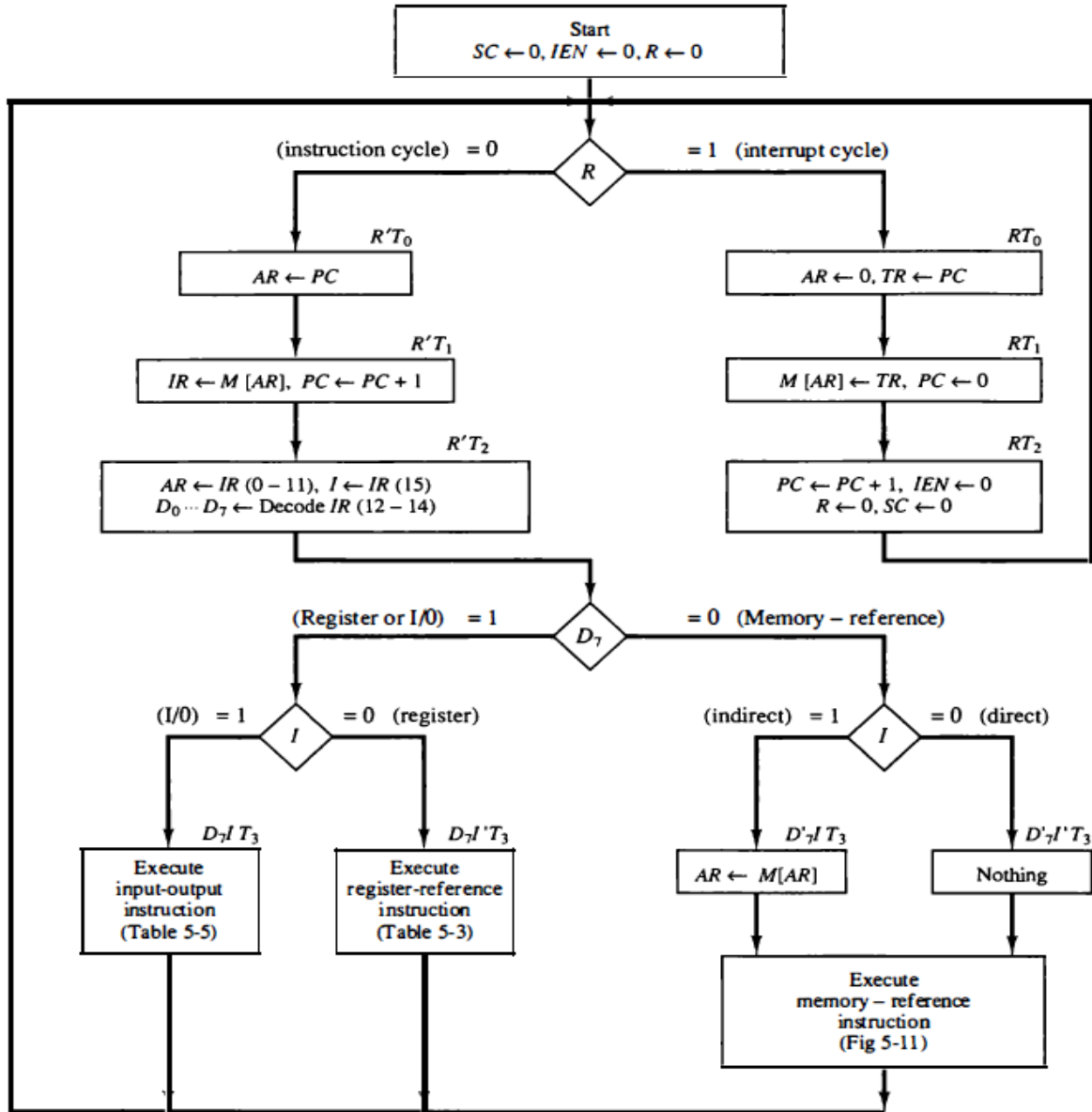
### 5.10 Temel Bilgisayarın Tasarımı

Temel bilgisayar özet olarak aşağıdaki donanım parçalarından oluşmaktadır:

- 4096 Kelimeli 1 bitlik Anabellek
- 9 adet Saklayıcı (AR, PC, DR, IR, TR, OTR, INPR, AC, SC)
- Dekoder (3X8: İşlem kod çözücü, 4X16 Zamnlama Kod çözücü)
- 7 adet bayrak (S:Başla, I: Doğrudan / Dolaylı, E:Elde, R:Kesilim, IEN: Kesilim izin, FGI: Giriş, FGO:Çıkış)
- 16 Bitlik BUS
- Kontrol ve Zamanlama devresi
- Toplama, VE, Kaydırma, Değilleme işlemi için ALU

Temel bilgisayar, tek AC'li bir yapıdır. Kesinti programı ile değiştirilmiş, komut fazı dış dünya ile ikili bilgi alışverişini sağlayan bir yapıyı sağlamıştır.

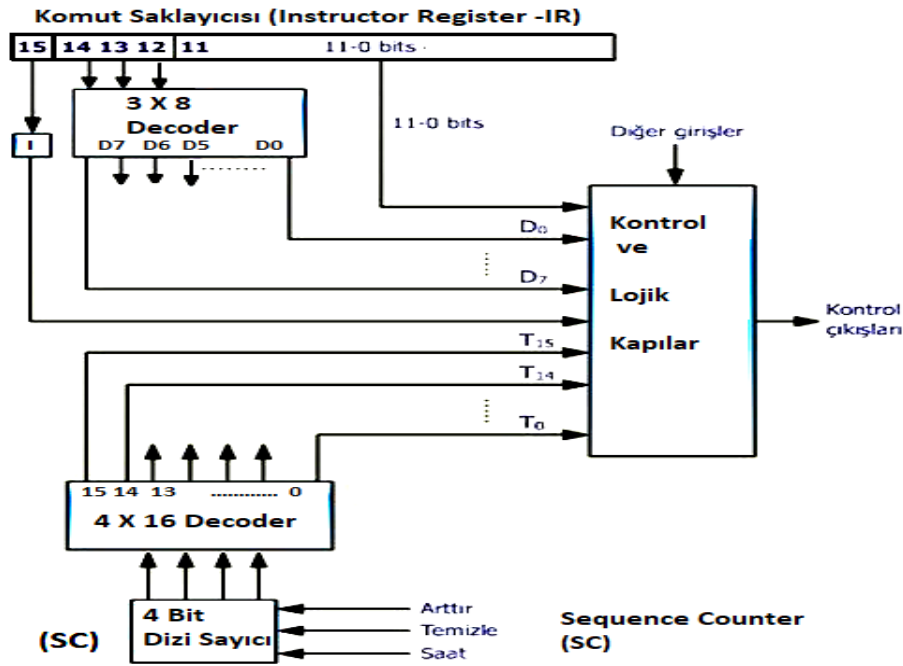
Temel bilgisayar için kesme döngüsü de dahil olmak üzere, komut döngüsünün son akış şeması aşağıdaki şekilde gösterilmektedir.



## 5.11 Kontrol Mantık Kapıları

Kontrol mantık kapılarının blok şeması daha önce tanımlanmıştı.





Bu kontrol devresinin

#### Girişleri:

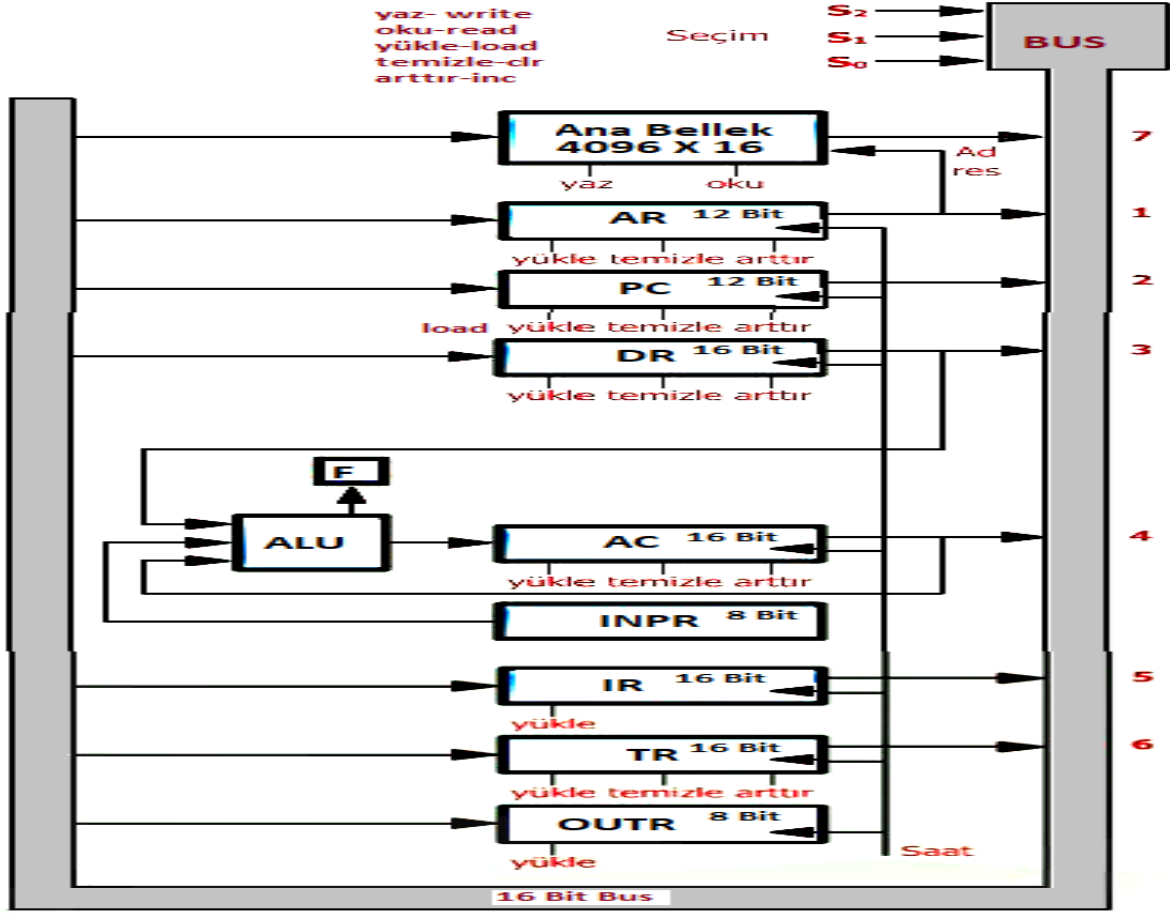
- 2adet kod çözücü (3X8, 4X16),
- I flip-floptan,
- IR'nin 0 ila 11 bitleri,
- AC = 0 olup olmadığını ve işaret bitini tespit etmek için AC nin 0 ila 15 arasındaki bitleri,
- DR = 0 olup olmadığını kontrol etmek için DR nin 0 ila 15 arasındaki bitleri,
- 7 adet Bayrak FF sini değerleri.

#### Çıkışları:

- Dokuz adet saklayıcının girişi için kontrol sinyali
- Ana bellek için okuma ve yazma sinyalleri
- Ffip-Flop lar için (Set, Clear ve Complement) sinyalleri
- Saklayıcıları seçmek için BUS kontrol devresinin (S1, S2, S3) sinyalleri
- AC saklayıcısı için Toplama ve Ve işlem sinyalleri

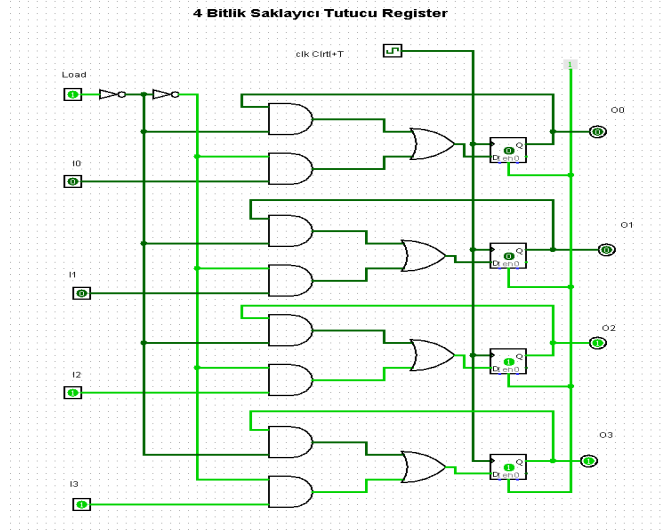
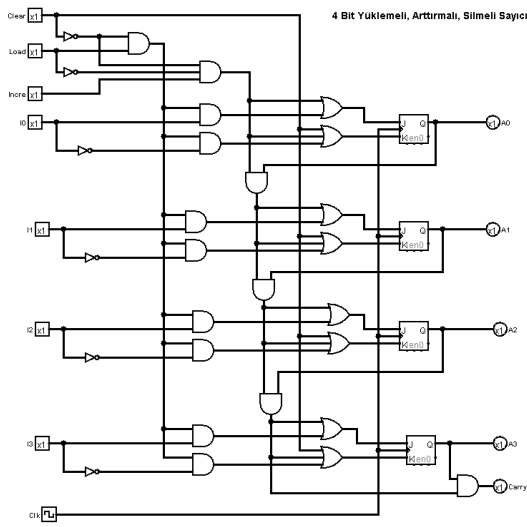
#### Saklayıcıların ve Anabelleğin Kontrolü

Ortak bir veri yolu sistemine (BUS) bağlı bilgisayarın saklayıcıların bağlantıları daha önce verilmişti.



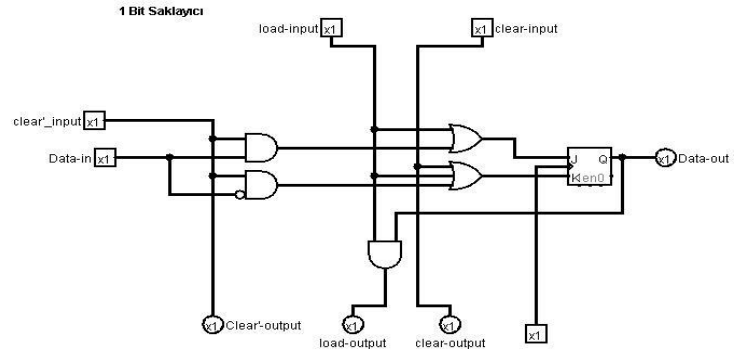
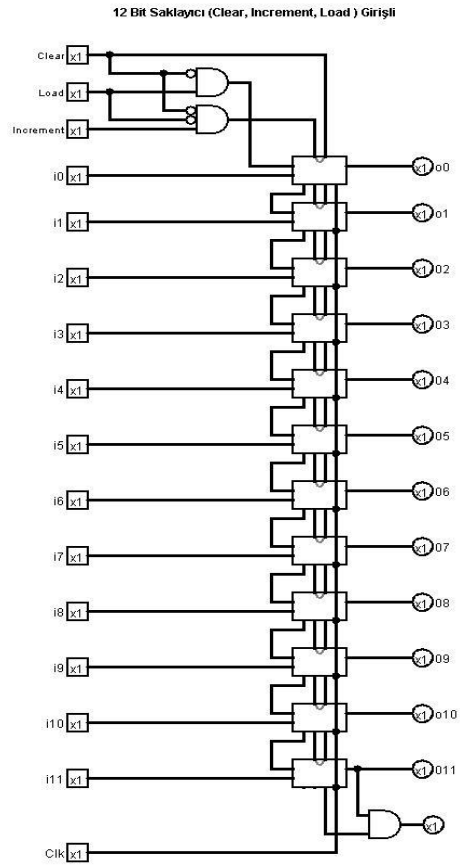
Saklayıcıların kontrol girişleri LD (yükle), INR (arttır), ve CLR (temizle) olarak tanımlanmıştı.

Bu şekilde girişlere sahip saklayıcıların yapıları da dah önce verilmişti.



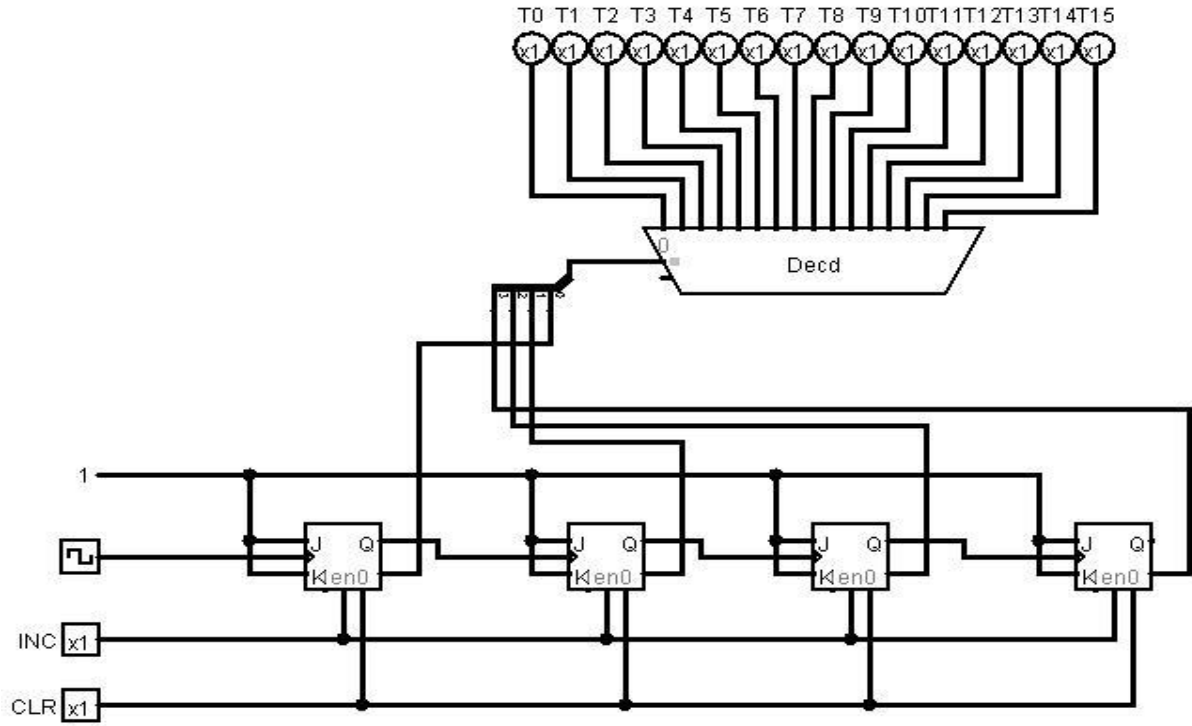
Yukarıda verilenlere göre Saklayıcıların, F-F ların, Anabelleğin, BUS kontrol devresinin ve ALU nun kontrol devrelerinin tasarımını yapalım;

Bunun için öncelikle (Clear, Increment, Load) girişli 1 bitlik bir saklayıcıyı modüler şekilde tasarlayıp, bunlardan 12 ve 16 bitlik saklayıcılar tasarlayalım



### 5.11.1 Zamanlama Kontrol İşareti:

Bilgisayar kontrol devresinin T0-T16 saat işaretlerini üretmek için gerekli olan SC (sequence control) ve 4X16 dekoder dan oluşan devre aşağıda verilmiştir. SC devresi clear girişine sahip olan asenkron 2 li sayıcı olarak tasarlanmıştır.



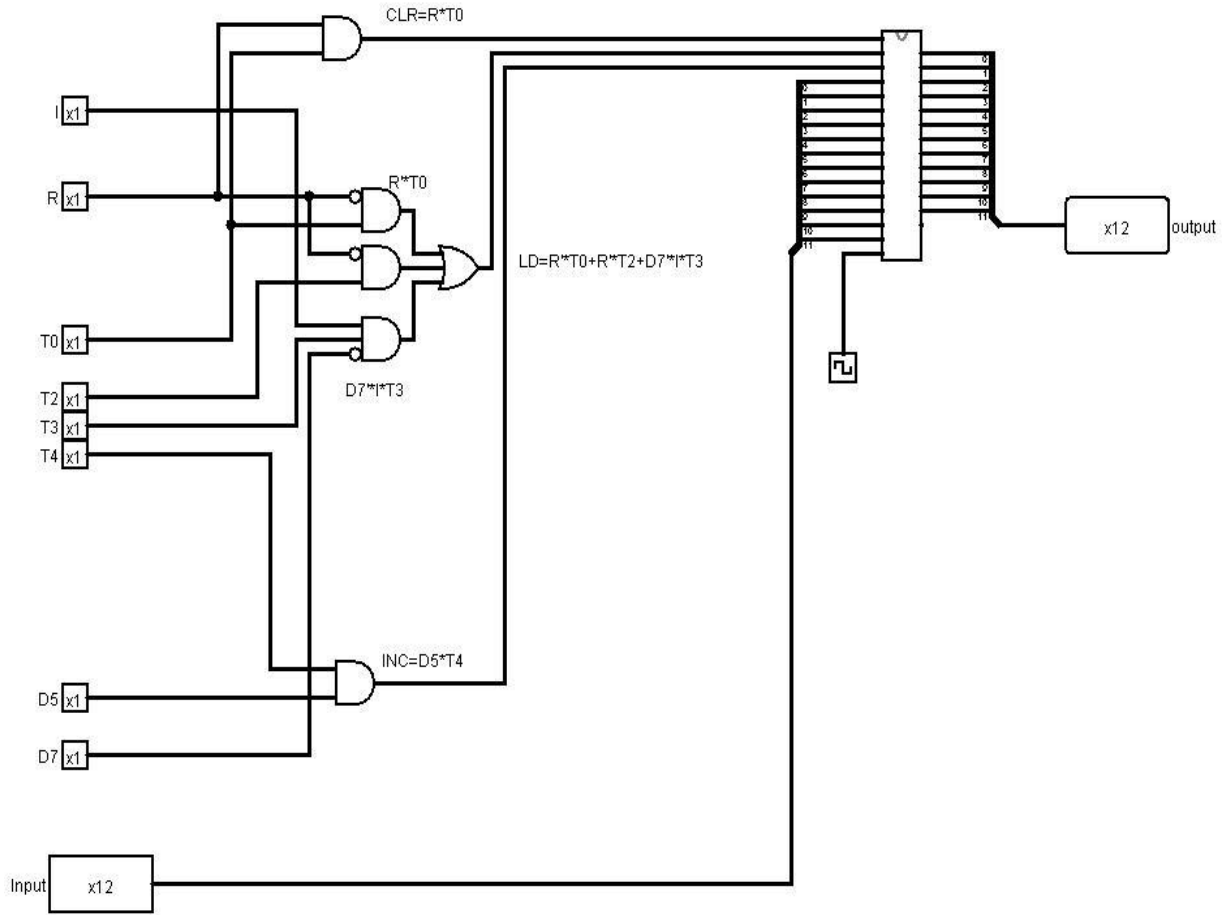
### 5.11.2 AR Saklayıcı:

12 bitlik bir saklayıcı olup Ana belleğin adresini tutar. Bu saklayıcı 12 bitlik Giriş/Çıkış bağlantı uçları ile BUS yapısına bağlıdır, saklayıcıyı kontrol etmek için (LD, INC ve CLR) girişleri kullanılır. Çıkış uçları Ana belleğin adresini göstermek amacıyla direk olarak Anabelleğin adres girişine bağlıdır. Buna ait kontrol devresini tasarlamak için komut tablosundaki ( $AR \leq$ )nin giriş olarak tanımlandığı ifadeleri çıkaralım;

Komut	Giriş
$R' * T_0 : AR \leq PC$	Yükle (LD)
$R' * T_2 : AR \leq IR(0-11)$	
$D_7' * I * T_3 : AR \leq M[AR]$	
$R * T_0 : AR \leq 0$	Temizle (CLR)
$D_5 * T_4 : AR \leq AR + 1$	Arttır (INC)

Yukarıdaki tablo 3 giriş sinyaline göre düzenlenirse;

Yükle (AR)	$R' T_0 + R' * T_2 + D_7' * I * T_3$
Temizle (AR)	$R * T_0$
Arttır (AR)	$D_5 * T_4$



### PC Saklayıcı:

12 bitlik bir saklayıcı olup işlenecek komutun adresini tutar. Bu saklayıcı 12 bitlik Giriş/Çıkış bağlantı uçları ile BUS yapısına bağlıdır, saklayıcıyı kontrol etmek için (LD, INC ve CLR ) girişleri kullanılır. Buna ait kontrol devresini tasarlamak için komut tablosundaki ( $PC \leq$  )nin giriş olarak tanımlandığı ifadeleri çıkaralım;

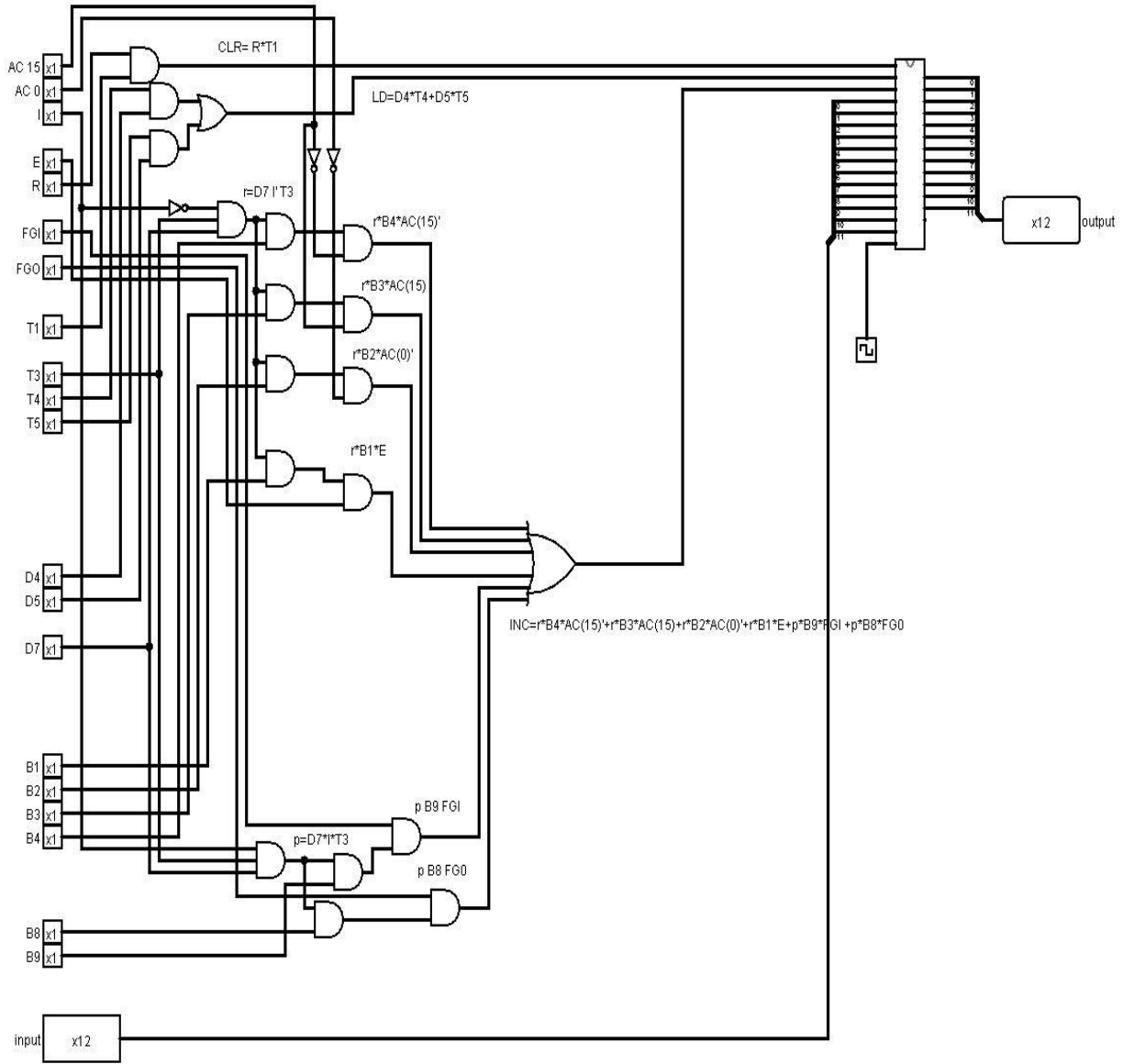
$$p = D7 * I * T3, \quad R = D7 * I * T3$$

Komut	Giriş
$D4 * T4 : PC \leq AR$	Yükle (LD)
$D5 * T5 : PC \leq AR$	
$R * T1 : PC \leq 0$	Temizle (CLR)
$R' * T1 : PC \leq PC + 1$	Arttır
$R * T2 : PC \leq PC + 1$	
$D6 * T6$ Eğer ( $DR = 0$ ) ise: $PC \leq PC + 1$	
$r * B4$ Eğer ( $AC(15) = 0$ ) ise: $PC \leq PC + 1$	
$r * B3$ Eğer ( $AC(15) = 1$ ) ise: $PC \leq PC + 1$	
$r * B2$ Eğer ( $AC = 0$ ) ise: $PC \leq PC + 1$	

r*B1 Eğer (E=1) ise: $PC \leq PC+1$	(INC)
p*B9 Eğer (FGI=1) ise: $PC \leq PC+1$	
p*B9 Eğer (FGO=1) ise: $PC \leq PC+1$	

Yukarıdaki tablo 3 giriş sinyaline göre düzenlenirse;

Yükle (PC)	$D4*T4 + D5*T5 :$
Temizle (PC)	$R'*T1$
Arttır (PC)	$R'*T1 + R*T2 +$ $(D6*T6 \text{ Eğer } (DR=0) \text{ ise}) +$ $(r*B4 \text{ Eğer } (AC(15)=0) \text{ ise}) +$ $(r*B3 \text{ Eğer } (AC(15)=1) \text{ ise}) +$ $(r*B2 \text{ Eğer } (AC=0) \text{ ise}) +$ $(r*B1 \text{ Eğer } (E=1) \text{ ise}) +$ $(p*B9 \text{ Eğer } (FGI=1) \text{ ise}) +$ $(p*B9 \text{ Eğer } (FGO=1) \text{ ise})$



### 5.11.3 DR Saklayıcı:

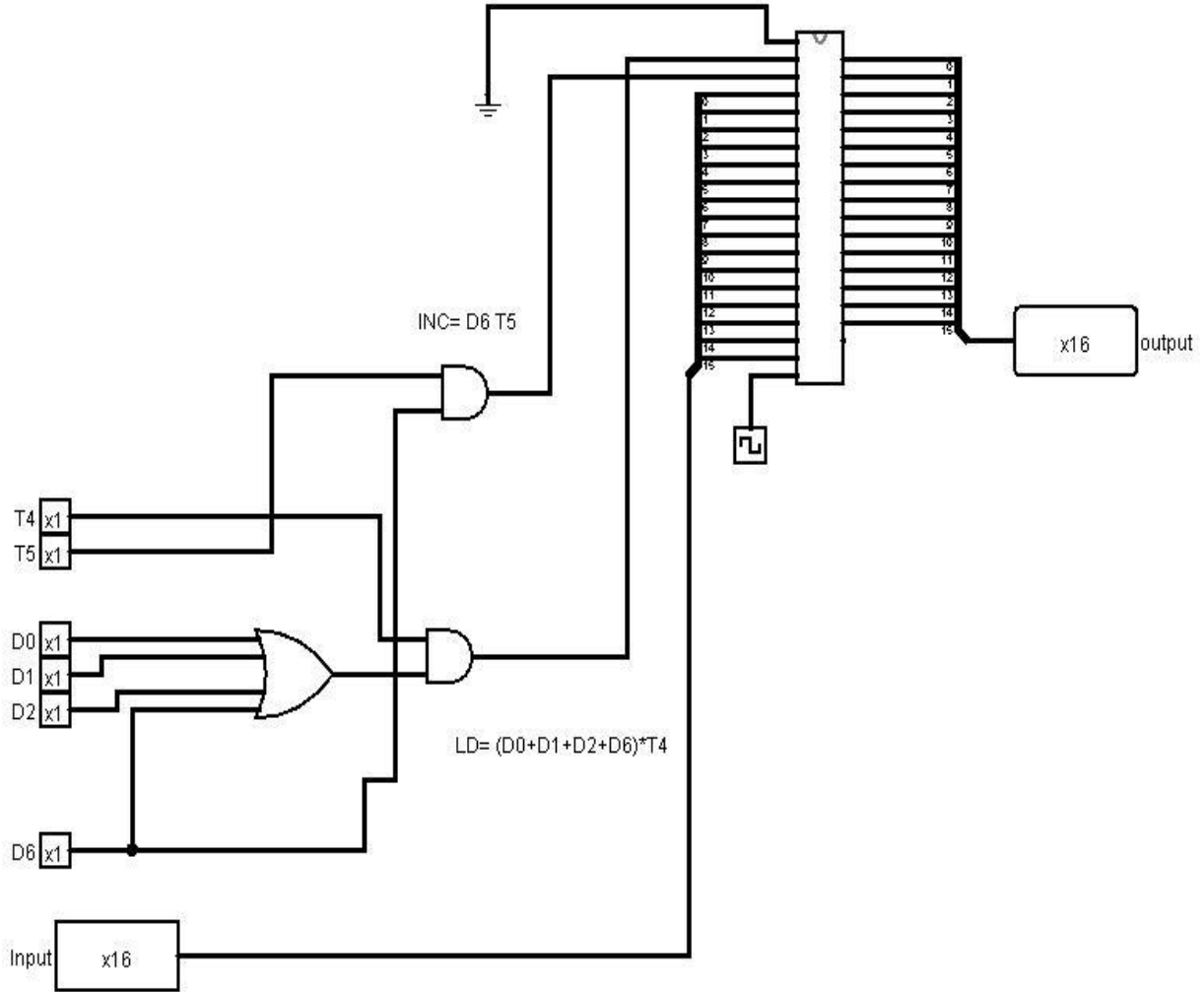
16 bitlik bir saklayıcı olup operantın Ana bellekteki değerini tutar. 16 bitlik Giriş/Çıkış bağlantı uçları ile BUS yapısına bağlıdır. Bu saklayıcıyı kontrol etmek için (LD ve INC) girişleri kullanılır. Buna ait kontrol devresini tasarlamak için komut tablosundaki (DR<= ) nin giriş olarak tanımlandığı ifadeleri çıkaralım;

Komut	Giriş
D0*T4 : DR<=M[AR]	Yükle (LD)
D1*T4: DR<=M[AR]	
D2*T4: DR<=M[AR]	
D6*T4: DR<=M[AR]	
D6*T5: DR<=DR+1	Arttır (INC)

Yukarıdaki tablo 2 giriş sinyaline göre düzenlenirse;



Yükle (PC)	$T4*(D0+ D1+ D2+ D6)$
Temizle (DR)	$D6*T5$

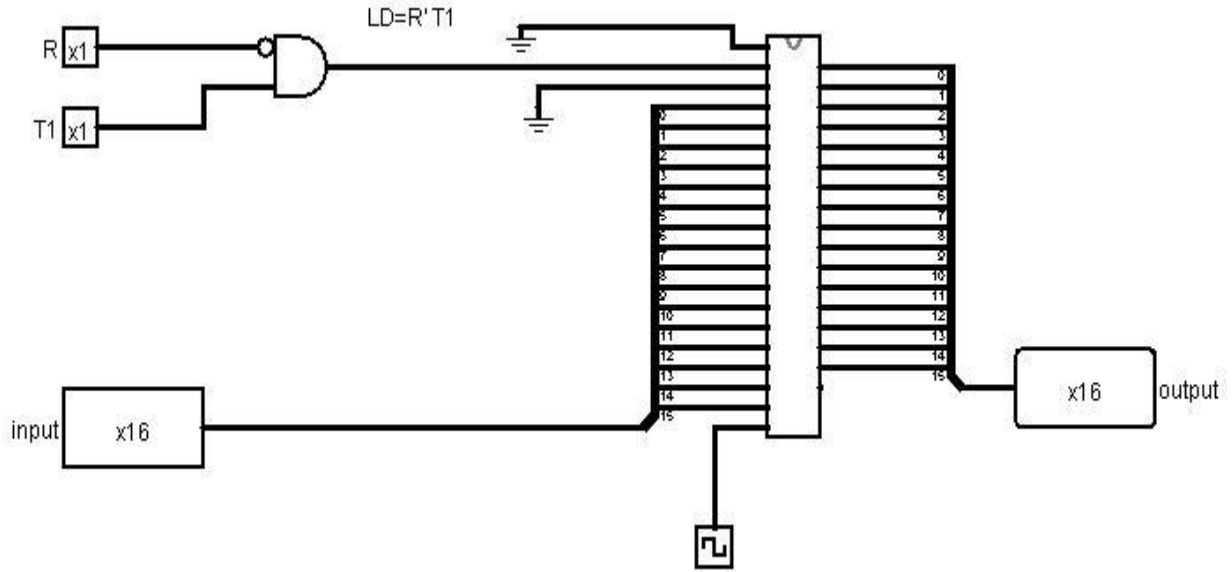


#### 5.11.4 IR Saklayıcı:

16 bitlik bir saklayıcı olup komutun kodunu tutar. 16	<b>Giriş</b>
$R*T1 : IR \leq M[AR]$	<b>Yükle (LD)</b>

Yukarıdaki tablo 1 giriş sinyaline göre düzenlenirse;

Yükle (IR)	$R'*T1$
------------	---------

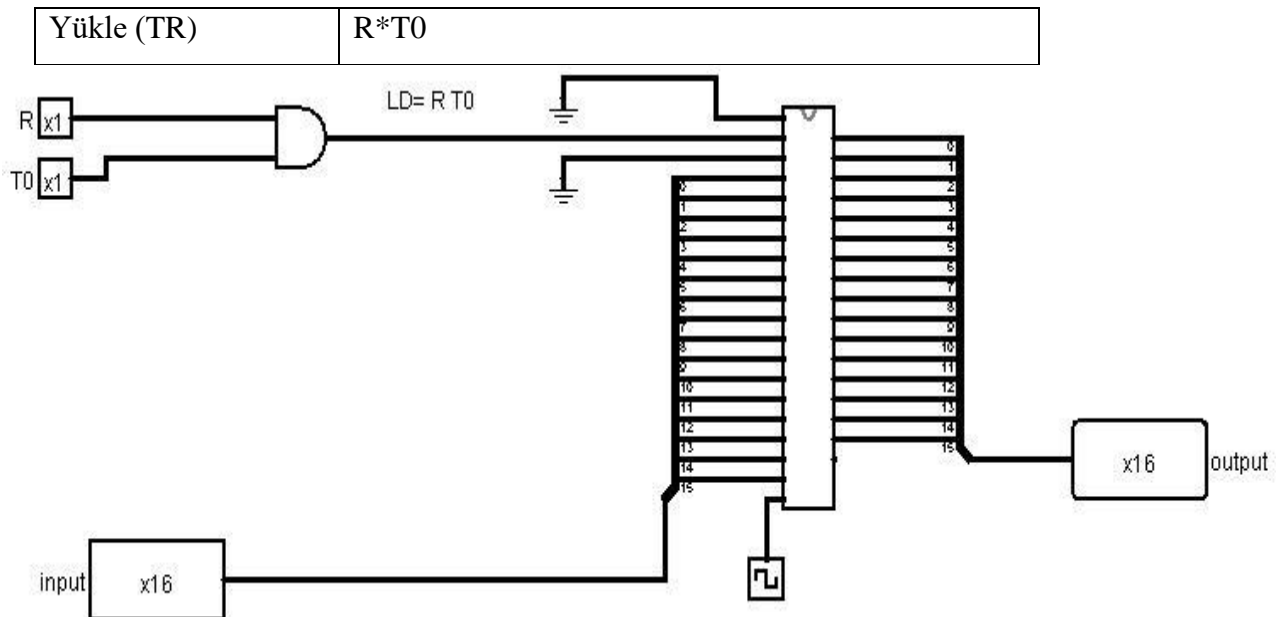


### 5.11.5 TR Saklayıcı:

16 bitlik bir saklayıcı olup geçici bilgi tutar. 16 bitlik Giriş/Çıkış bağlantı uçları ile BUS yapısına bağlıdır. Bu saklayıcıyı kontrol etmek için (LD) girişi kullanılır. Buna ait kontrol devresini tasarlamak için komut tablosundaki ( $TR \leq$ ) nin giriş olarak tanımlandığı ifadeleri çıkaralım;

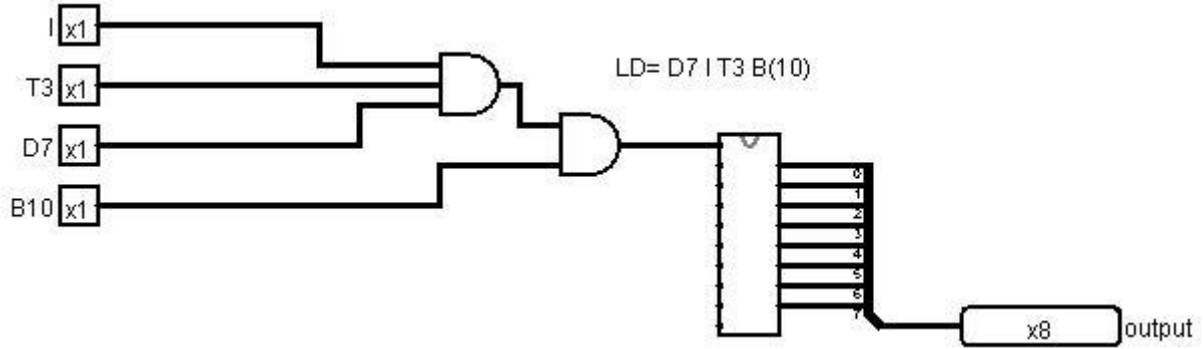
Komut	Giriş
$R * T0 : TR \leq PC$	Yükle (LD)

Yukarıdaki tablo 1 giriş sinyaline göre düzenlenirse;



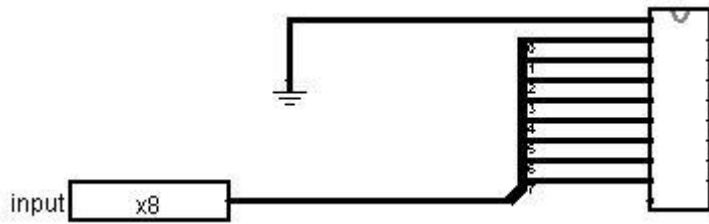
### 5.11.6 OUTF Saklayıcı:

8 bitlik bir saklayıcı olup 8 bitlik Giriş bağlantı uçları ile BUS yapısına bağlıdır. Bu saklayıcıyı kontrol etmek için (LD) girişleri kullanılır. Buna ait kontrol devresini tasarlamak için komut tablosundaki OUTF nin giriş olarak tanımlandığı ifadeleri çıkaralım;



### 5.11.7 INPR Saklayıcı:

8 bitlik bir saklayıcı olup 8 bitlik Giriş bağlantı uçları ile AC girişine direk olarak bağlıdır. İlave bir kontrol devresi tasarımı gerekmez.



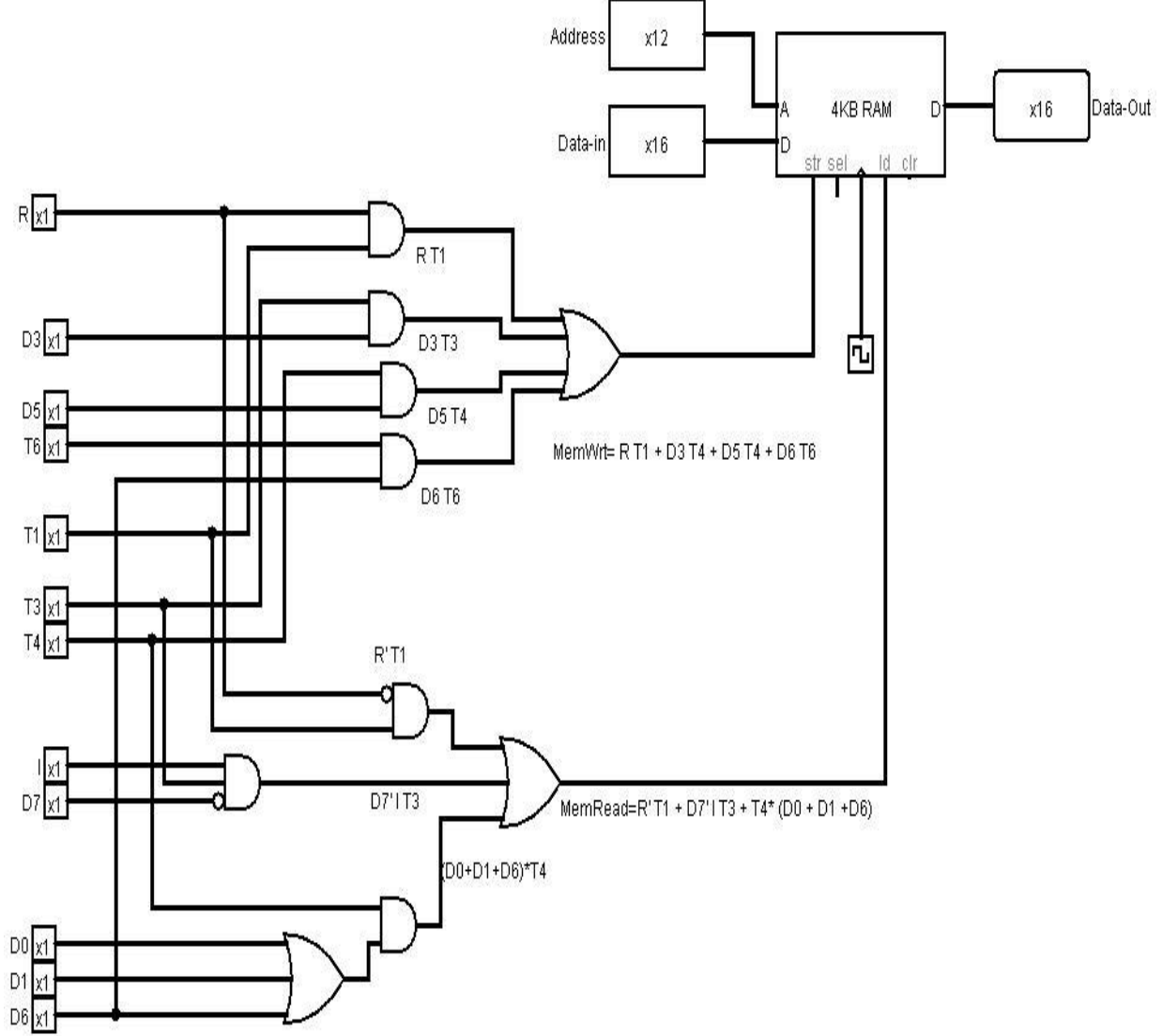
### 5.11.8 Memory Read, Memory Write:

Devremizde 8 K Word lük bir ana bellek (RAM) ünitesi kullanılmış olup 12 bitlik adres ünitesi direkt olarak AR saklayıcısına bağlanmıştır. Data uçları ise 16 bit üzerinden BUS a bağlıdır. Ana belleğin okuma sinyali ( $\leq M[AR]$ ) ve yazma sinyali ( $M[AR] \leq$ ) ait eşitlikler aşağıda verilmiştir.

OKUMA (MEMR)	YAZMA (MEMW)
R'*T1: IR $\leq$ M[AR]	R*T1 : M[AR] $\leq$ TR
D7'*I*T3: AR $\leq$ M[AR]	D3*T4: M[AR] $\leq$ AC
D0*T4: DR $\leq$ M[AR]	D5*T4: M[AR] $\leq$ PC
D1*T4: DR $\leq$ M[AR]	D6*T6: M[AR] $\leq$ DR
D6*T4: DR $\leq$ M[AR]	
D2*T4: DR $\leq$ M[AR]	

Yukarıdaki tablo Memr ve Memw sinyallerine göre düzenlenirse;

<b>Memr</b>	$R * T1 + D7 * I * T3 + T4 * (D0 + D1 + D6 + D2)$
<b>Memw</b>	$R * T1 + D3 * T4 + D5 * T4 + D6 * T6$



### 5.11.9 AC Saklayıcı:

16 bitlik bir saklayıcı olup Akümülatör görevini yapar. Giriş data uçları 16 bit üzerinden ALU ya , çıkış data uçları BUS yapısına bağlıdır. Bu saklayıcıyı kontrol etmek için (LD, INC, CLR) girişleri kullanılır. Buna ait kontrol devresini tasarlamak için komut tablosundaki (AC<= ) nin giriş olarak tanımlandığı ifadeleri çıkaralım;

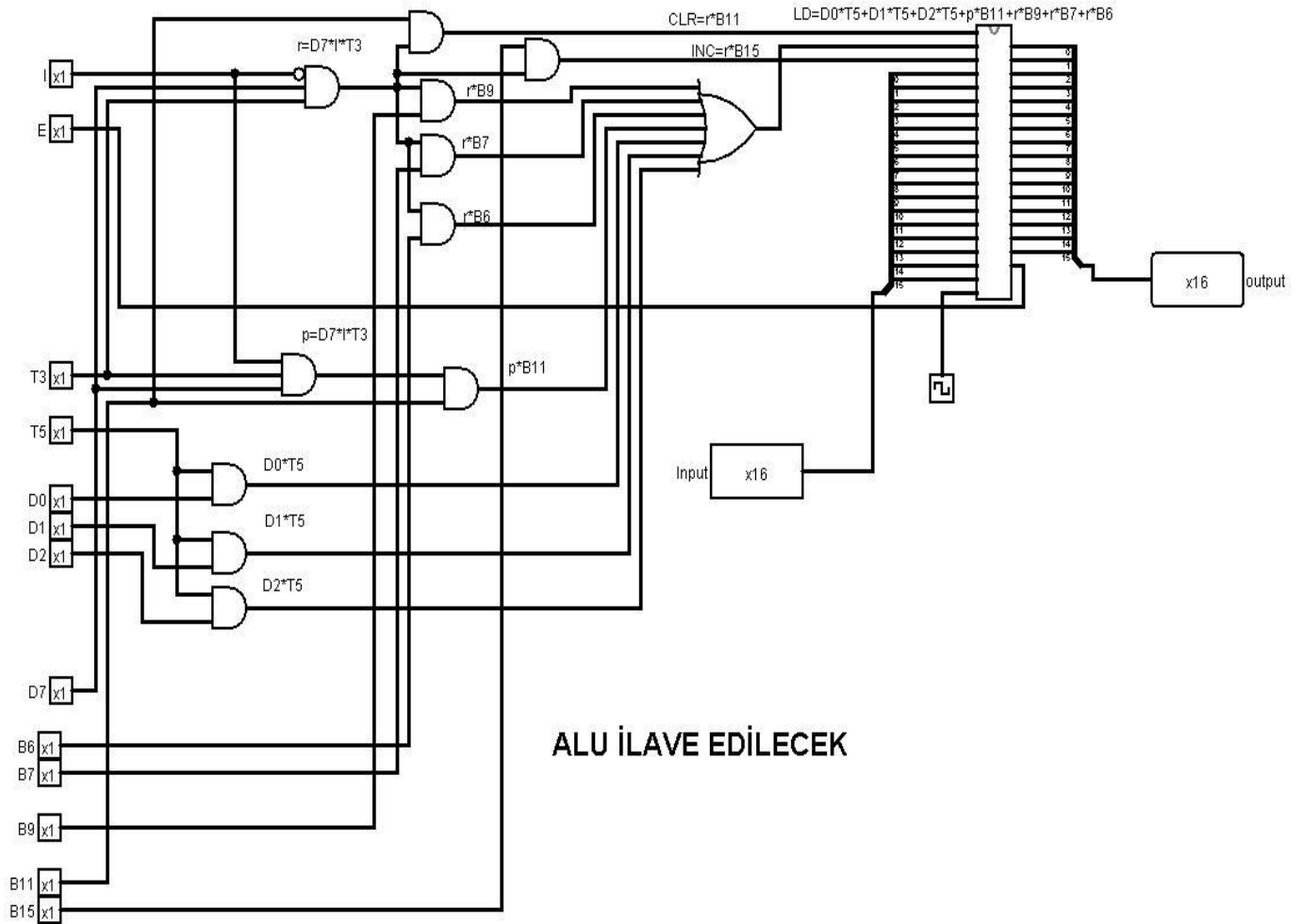
$$p = D7 * I * T3, \quad R = D7 * I * T3$$

Komut	Giriş
D0*T5: AC<=AC^DR ; AND	Yükle (LD)
D1*T5: AC<=AC+DR ; ADD	

D2*T5: $AC \leq DR$ ; DR	
r*B9: $AC \leq AC'$ ; COMPLEMENT AC	
r*B7: $AC \leq shr(AC) + AC(15) \leq E$ ; SHR AC	
r*B6: $AC \leq shl(AC) + AC(0) \leq E$ ; SHL AC	
p*B11: $AC(0-7) \leq INPR$ ; INPR	
r*B11 : $AC \leq 0$	Temizle (CLR)
r*B5 : $AC \leq AC+1$	Arttır (INC)

Yukarıdaki tablo 3 giriş sinyaline göre düzenlenirse;

Yükle (AC)	$D0*T5 + D1*T5 + D2*T5 + r*B9 + r*B7 + r*B6 + p*B11$
Temizle (AC)	$r*B11$
Arttır (AC)	$r*B5$

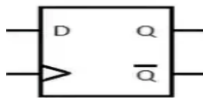
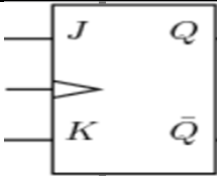


### 5.11.10 Bayraklar

Bayraklar birer Flip-Flop olup kontrol ifadeleri aşağıda çıkarılmıştır.

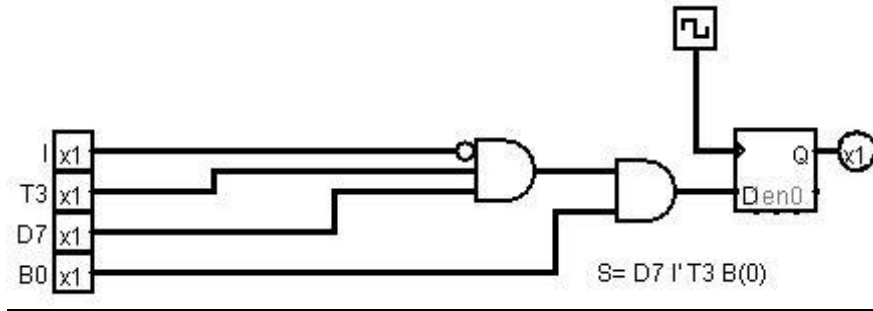
#### *Hatırlatma;*

D FF

D	Q		
0	0		
1	1		
JK FF			
J	K	Q	
0	0	Q	
0	1	0	
1	0	1	
1	1	Q'	

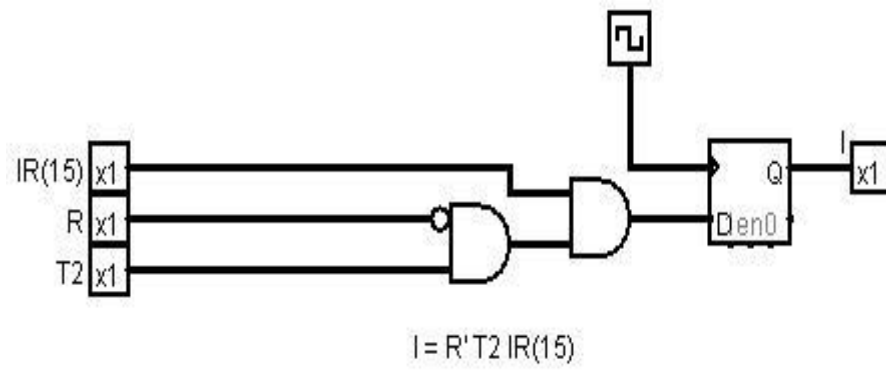
**S FLAG:** (HALT) Durdurma bayrağı olup lojik ifadesi ;

$r*B0: S \leq 0$



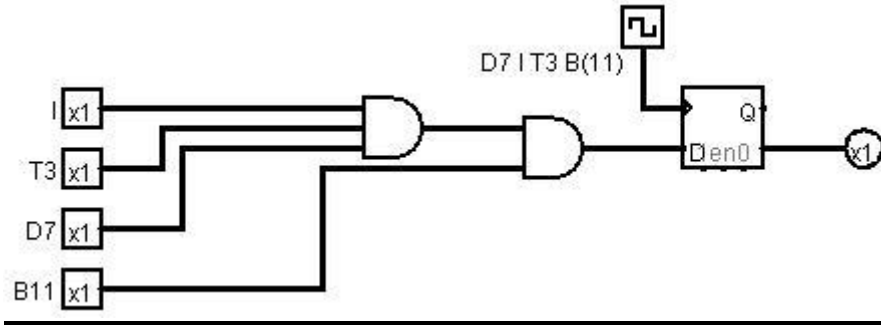
**I FLAG:** (INTERRUPT) Kesilim bayrağı olup lojik ifadesi ;

$R'T2:I \leq IR(15)$



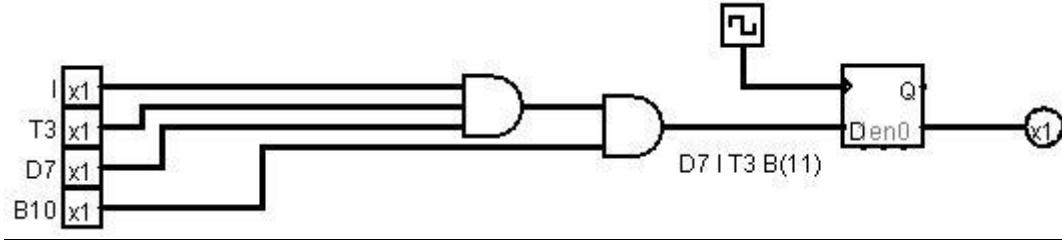
**FGI FLAG :** (INPUT FLAG) Seri port giriş durum bayrağı olup lojik ifadesi ;

p\*B11: FGI<=0



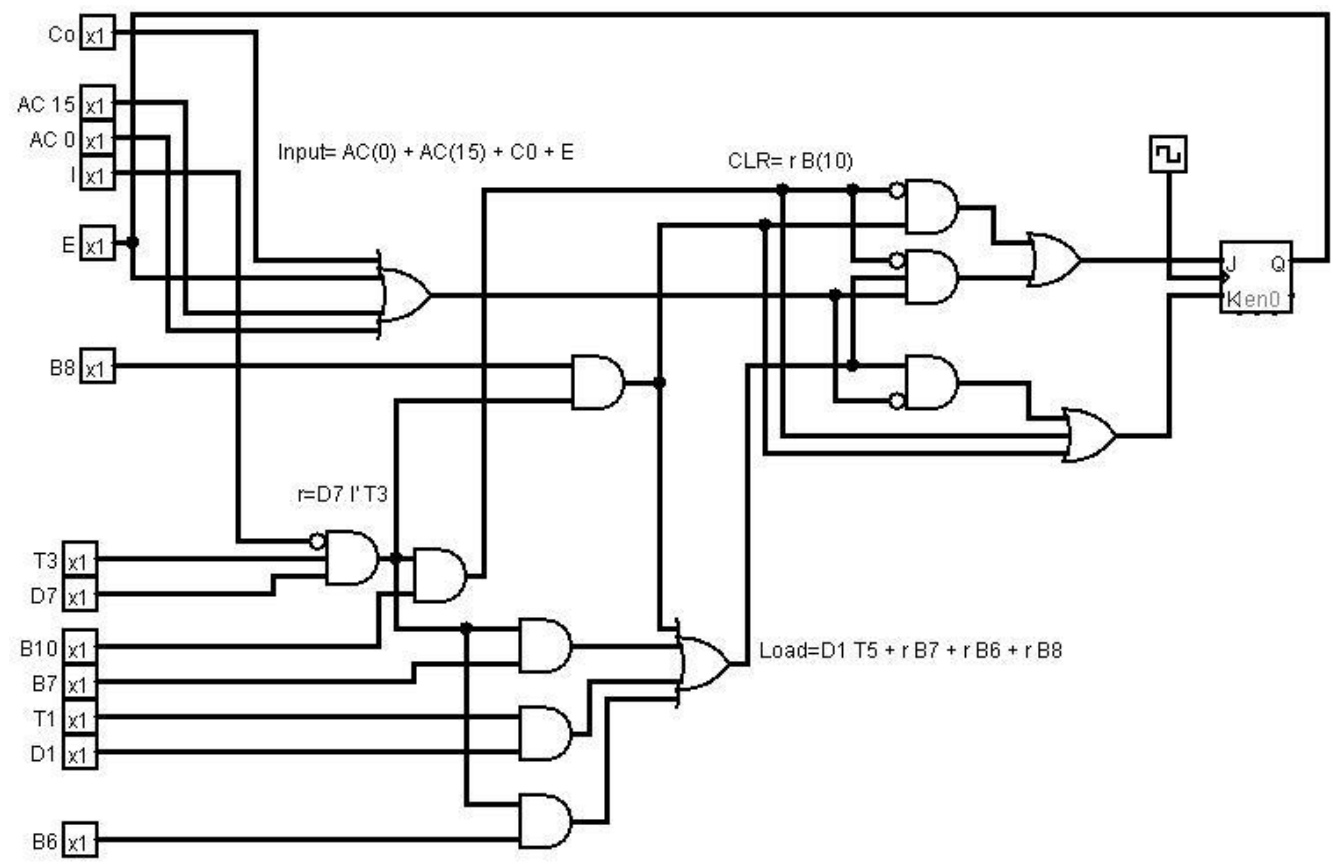
**FGO FLAG** (OUTPUT FLAG) Seri port çıkış durum bayrağı olup lojik ifadesi ;

p\*B10: FG0<=0



**E FLAG** (CARRY FLAG) Elde bayrağı olup lojik ifadesi ;

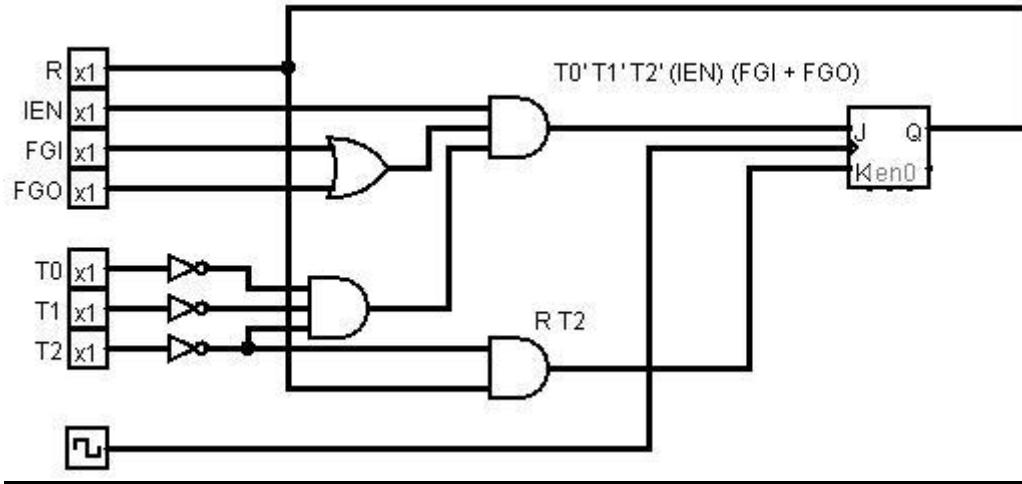
D1*T5: E <=Co
r*B11: E <=0
r*B8: E <=E'
r*B7: E <=AC(0)
r*B6: E <=AC(15)





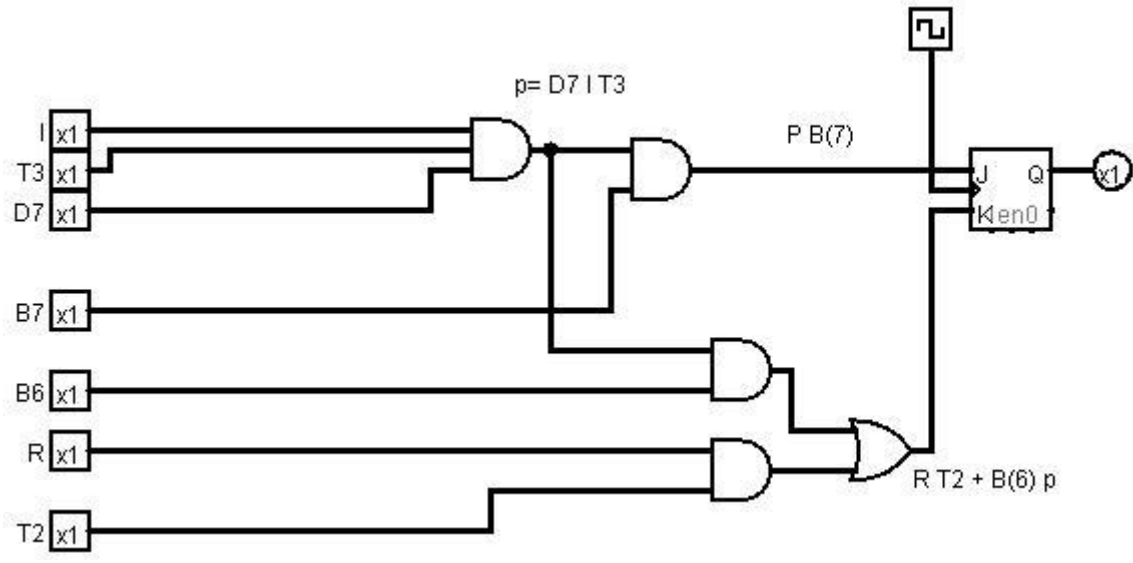
***R FLAG*** (INTERRUPT STATE FLAG) Kesilim durum bayrağı olup lojik ifadesi ;

$T0' * T1' * T2' * (IEN) * (FGI + FGO) : R \leq 1$
$R * T2 : R \leq 0$



***IEN FLAG***: (INTERRUPT ENABLE FLAG) Kesilim etkinleştirme bayrağı olup lojik ifadesi ;

$R * T2 : IEN \leq 0$
$p * B6 : IEN \leq 0$
$p * B7 : IEN \leq 1$

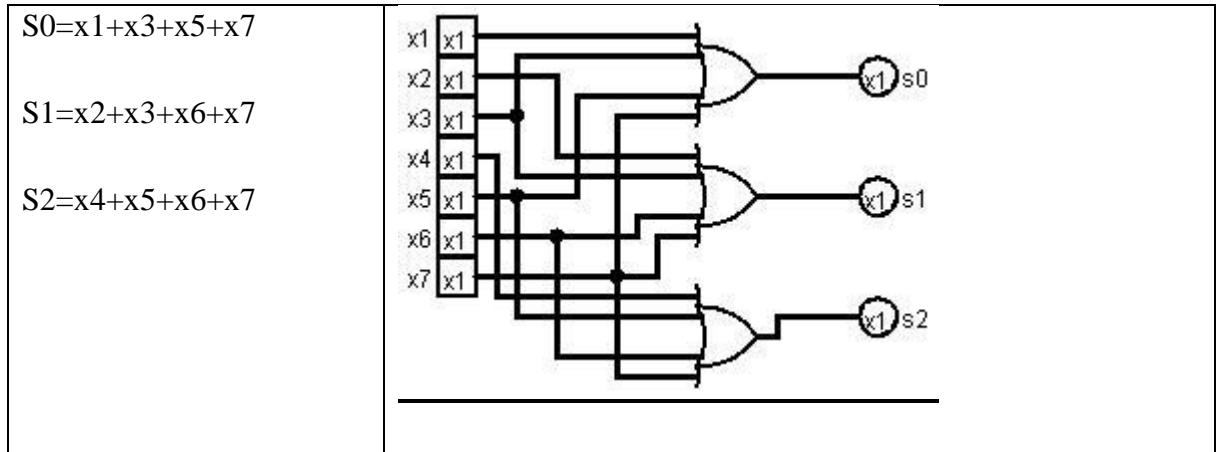


## 5.12 BUS KONTROL

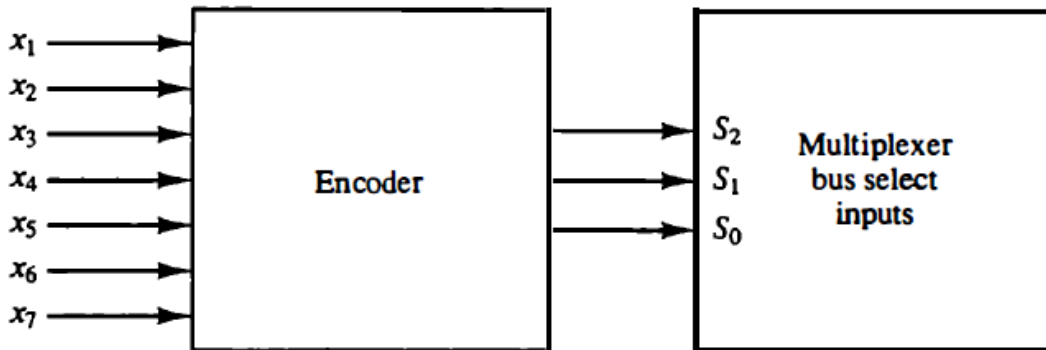
Şekil gösterilen 16 bitlik ortak veri yolu (BUS), S2, S1 ve S0 seçim girişleri tarafından kontrol edilir. Her veri yolu girişi ile gösterilen ondalık sayı, ilgili saklayıcıyı seçmek için seçim girişlerine uygulanması gereken eşdeğer ikili sayıyı gösterir. Aşağıda verilen tablo ilgili saklayıcıyı seçmek için gerekli giriş değerlerini gösterir.

Giriş							Çıkış			Saklayıcı
x1	x2	x3	x4	x5	x6	x7	S2	S1	S0	
0	0	0	0	0	0	0	0	0	0	Yok
1	0	0	0	0	0	0	0	0	1	AR
0	1	0	0	0	0	0	0	1	0	PC
0	0	1	0	0	0	0	0	1	1	DR
0	0	0	1	0	0	0	1	0	0	AC
0	0	0	0	1	0	0	1	0	1	IR
0	0	0	0	0	1	0	1	1	0	TR
0	0	0	0	0	0	1	1	1	1	Ana Bellek

Bu amaçla gerekli olan 7X3 lük kodlayıcıyı yapısı ve mantıksal ifadesi aşağıda verilmiştir.



Bu şekilde aşağıdaki kontrol devresi elde edilecektir.



Her enkoder girişı için mantığı belirlemek için, ilgili kaydı veri yoluna yerleştiren kontrol fonksiyonlarını bulmak gerekir. Örnek olarak  $x1=1$  yapmak için ilgili logic devreyi tasarlayabilmek için AR 'nin kaynak olarak kullanıldığı ( $\leq AR$ ) komut dizilerini bulmak gerekir. Burdada  $x1$  girişı AR yi seçtiğı için AR dikkate alınmıştır. Herbir giriş değeri için ilgili saklayıcının kaynak olarak kullanıldığı saklayıcı seçilmelidir.

### **X1 ( $\leq AR$ )**

D4*T4 : PC $\leq$ AR
D5*T5 : PC $\leq$ AR

Giriş değerine göre düzenlenirse;

$X1 = D4*T4 + D5*T5$
----------------------

### **X2 ( $\leq PC$ )**

R'*T0 : AR $\leq$ PC
D5*T4 : M[AR] $\leq$ PC
R*T0 : AR $\leq$ PC

Giriş değerine göre düzenlenirse;

$X2 = R'*T0 + D5*T4 + R*T0$
-----------------------------

### **X3 ( $\leq DR$ )**

D2*T5 : AC $\leq$ DR
D6*T6 : M[AR] $\leq$ DR

Giriş değerine göre düzenlenirse;

$X3 = D2*T5 + D6*T6$
----------------------

### **X4 ( $\leq AC$ )**

D3*T4 : M[AR] $\leq$ AC
-------------------------

Giriş değerine göre düzenlenirse;

$$X4 = D3 * T4$$

**X5 (<=IR)**

$$R' * T2 : AR \leq IR$$

Giriş değerine göre düzenlenirse;

$$X5 = R' * T2$$

**X6 (<=TR)**

$$R * T1 : M[AR] \leq TR$$

Giriş değerine göre düzenlenirse;

$$X6 = R * T1$$

**X7 (<=M[ ])**

$$R' * T1 : \leq M[AR]$$

$$D7 * I * T3 : \leq M[AR]$$

$$D0 * T4 : \leq M[AR]$$

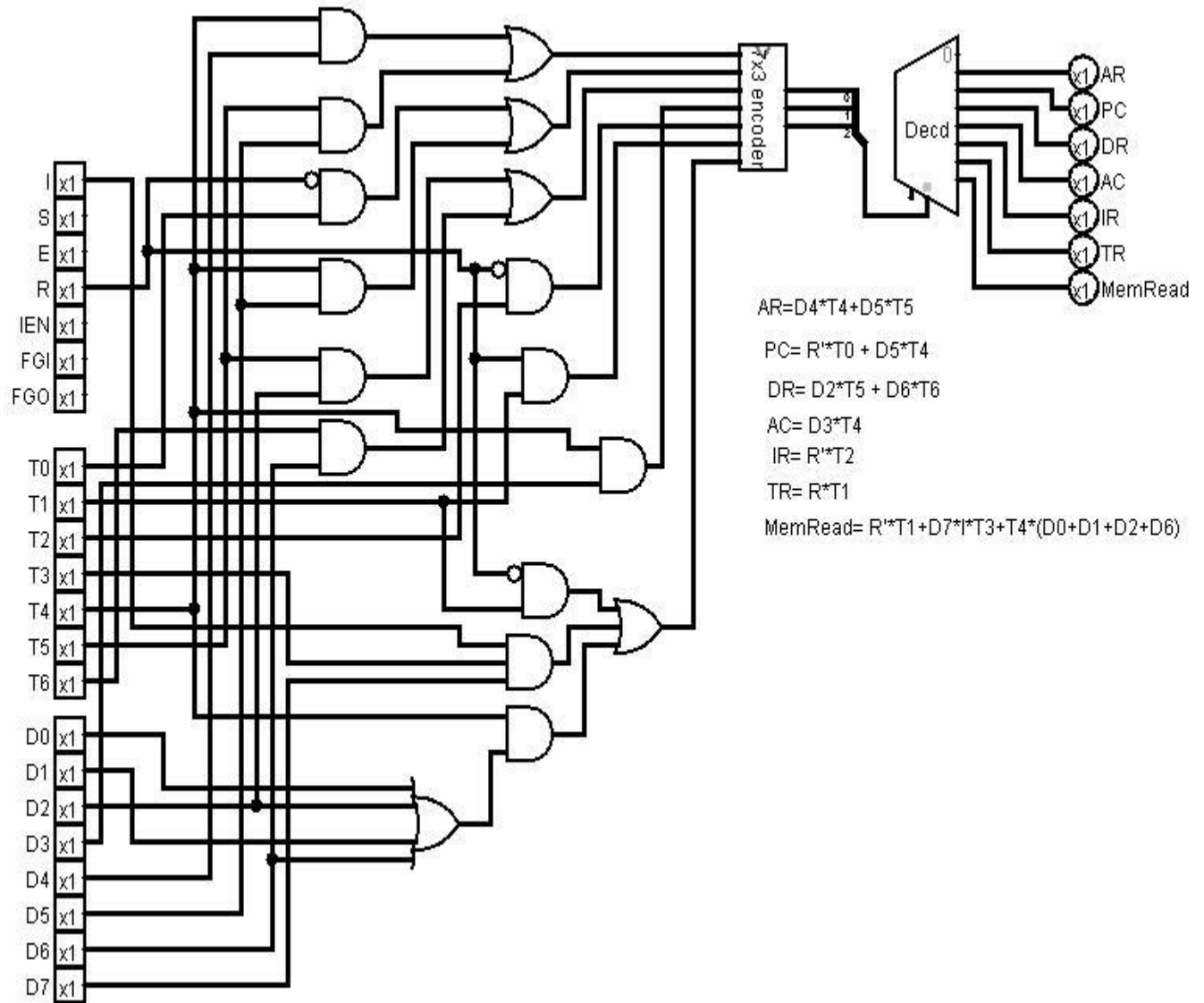
$$D1 * T4 : \leq M[AR]$$

$$D2 * T4 : \leq M[AR]$$

$$D6 * T4 : \leq M[AR]$$

Giriş değerine göre düzenlenirse;

$$X7 = R' * T1 + D7 * I * T3 + T4 * (D0 + D1 + D2 + D6)$$



Sembol	Koşul	İşlem
Fetch	$R' T_0$	$AR \leftarrow PC, SC \leftarrow SC+1$
	$R' T_1$	$IR \leftarrow M[AR], PC \leftarrow PC+1, SC \leftarrow SC+1$
Decode	$R' T_2$	$(D_0 \dots D_7) \leftarrow \text{Decode } IR(12-14), AR \leftarrow IR(0-11), I \leftarrow IR(15), SC \leftarrow SC+1$
Indirect	$D_7' I T_3$	$AR \leftarrow M[AR], SC \leftarrow SC+1$
Interrupt	$T_0' T_1' T_2' (IEN)$ $(FGI+FGO)$	$R \leftarrow 1$
	$R T_0$	$AR \leftarrow 0, TR \leftarrow PC, SC \leftarrow SC+1$
	$R T_1$	$M[AR] \leftarrow TR, PC \leftarrow 0, SC \leftarrow SC+1$
	$R T_2$	$PC \leftarrow PC+1, IEN \leftarrow 0, R \leftarrow 0, SC \leftarrow 0$
Memory Addressing Mode		
AND (And accumulator)	$D_0 T_4$	$DR \leftarrow M[AR], SC \leftarrow SC+1$
	$D_0 T_5$	$AC \leftarrow AC \wedge DR, SC \leftarrow 0$

<b>ADD</b> (Add accumulator)	D <sub>1</sub> T <sub>4</sub>	DR ← M[AR] , SC ← SC+1
	D <sub>1</sub> T <sub>5</sub>	AC ← AC + DR, E ← Co, SC ← 0
<b>LDA</b> (Load accumulator)	D <sub>2</sub> T <sub>4</sub>	DR ← M[AR] , SC ← SC+1
	D <sub>2</sub> T <sub>5</sub>	AC ← DR, SC ← 0
<b>STA</b> (Store accumulator)	D <sub>3</sub> T <sub>4</sub>	M[AR] ← AC, SC ← 0
<b>BUN</b> (Branch Uncontionally)	D <sub>4</sub> T <sub>4</sub>	PC← AR, SC ← 0
<b>BSA</b> (Branch and Save Return Address)	D <sub>5</sub> T <sub>4</sub>	M[AR] ← PC, SC ← SC+1
	D <sub>5</sub> T <sub>5</sub>	PC ← AR, SC ← 0
<b>ISZ</b> (Increment and Skip if Zero)	D <sub>6</sub> T <sub>4</sub>	DR ← M[AR] , SC ← SC+1
	D <sub>6</sub> T <sub>5</sub>	DR ← DR +1, SC ← SC+1
	D <sub>6</sub> T <sub>6</sub>	M[AR] ← DR, Eğer (DR=0) ise (PC← PC+1) , SC ← 0
<b>Register Addressing Mode</b> (r= D <sub>7</sub> I' T <sub>3</sub> ), IR(i)=B, (i=0, 1, 2, 3, ...11)		
<b>CLA</b> (Clear accumulator)	r B <sub>11</sub>	AC←0, SC←0
<b>CLE</b> (Clear Cary Flag)	r B <sub>10</sub>	E←0, SC←0
<b>CMA</b> (Complement accumulator)	r B <sub>9</sub>	AC← AC', SC←0
<b>CME</b> (Complement Cary Flag)	r B <sub>8</sub>	E← E', SC←0
<b>CIR</b> (Circulate Right Accumulator)	r B <sub>7</sub>	AC←shr AC, AC(15) ← E, E←AC(0), SC←0
<b>CIL</b> (Circulate Left Accumulator)	r B <sub>6</sub>	AC←shl AC, AC(0) ← E, E←AC(15)
<b>INC</b> (Increment Accumulator)	r B <sub>5</sub>	AC← AC+1, SC←0
<b>SPA</b> (Skip if Positive)	r B <sub>4</sub>	Eğer (AC(15)=0) ise (PC←PC+1), SC←0
<b>SNA</b> (Skip if Negative)	r B <sub>3</sub>	Eğer (AC(15)=1) ise (PC←PC+1), SC←0
<b>SZA</b> (Skip if A C zero)	r B <sub>2</sub>	Eğer (AC=0) ise (PC←PC+1), SC←0
<b>SZE</b> (Skip if E zero)	r B <sub>1</sub>	Eğer (E=1) ise (PC←PC+1), SC←0
<b>HLT</b> (Halt)	r B <sub>0</sub>	S←0 (S başlangıç FF sidir.), SC←0
<b>Input/Output Addressing Mode</b> p=D <sub>7</sub> I T <sub>3</sub>		
<b>INP</b> (Input Character)	p B <sub>11</sub>	AC(0-7) ←INPR, FGI←0, SC←0
<b>OUT</b> (Output character)	p B <sub>10</sub>	OUTR← AC (0-7), FG0←0, SC←0
<b>SKI</b> (Skip on input flag)	p B <sub>9</sub>	Eğer(FGI=1) ise (PC←PC+1), SC←0
<b>SKO</b> (Skip on output flag)	p B <sub>8</sub>	Eğer(FGO=1) ise (PC←PC+1), SC←0
<b>ION</b> (Interrupt enable on)	p B <sub>7</sub>	IEN← 1, SC←0
<b>IOF</b> (Interrupt enable off)	p B <sub>6</sub>	IEN← 0, SC←0

**M=Memory, R=Interrupt Flag, IEN=Interrupt Enable Flag, FGI=Input Flag, FGO=Output Flag, S=Reset Flag**

**AC=Accumulator, AR=Address Register, DR=Data Register, IR=Instruction Register, TR= Temporary Register, INPR=Input Register, P=Program Counter, OUTR=Output Register**

## 6 TEMEL BİLGİSAYARIN PROGRAMLAMASI

“Yüksek seviyeli” (high level) programlar C, Fortran, Pascal, C++, Java, Pearl, vb. dilinde yazılmış, dolayısıyla, kullanıcıya daha yardımcı olan programlardır. Yüksek seviyeli bir dilde yazılan programı koşturabilmek için “koşturulabilen” (executable) programın veya ikili kodun (makine kodu) elde edilmesi gerekir. Bunun için derleyiciler (compiler) mevcuttur. Örneğin, C derleyicisi ile C programlama dilinde yazılmış olan bir programı koşturulabilecek formata, yani ikili makina diline döndürmek mümkün olur.

Assembler dili belli işlemleri yapmaya yarayan “sembol” lerden oluşmaktadır ve alt seviyeli bir dil grubudur. Bilgisayarın işlemcisine bağlı olarak, yapımçı firma tarafından tasarım sırasında belirlenmiştir. Makina dilindeki 0 ve 1'lerden oluşan dizileri hatırlamak veya onaltılı kodlar üzerinde çalışmak yerine, assembler dili ile çalışmak çok daha kolay olmaktadır. Assembler dilinde sembolik komutlarla yazılmış bir program ise “assembler” vasıtasıyla makina diline çevrilmektedir. Elde edilen ikili kodlar, bilgisayar işlemcisinde koşturularak sonuçlar elde edilir ve gösterilir. Aşağıda assembler yoluyla makina kodu elde edilmesi gösterilmiştir.

### 6.1 Temel Bilgisayarın Assembler Komutları

Bir önceki bölümde verilen tabloda temel bilgisayarda kullanılan tüm assembler komutları gösterilmiştir. Bu komutlar akılda tutulması çok daha kolay sembollerdir.

### 6.2 İkili Kod ve Assembler Kodu Arasındaki İlişki

İkili kod, onaltılı kod ve assembler kod arasındaki ilişkiyi göstermek için örnek olarak iki komutun üç koddaki değerleri aşağıda gösterilmiştir. Bilgisayar gösterilimi yerine “STA 012” ve “LDA 002” komutlarını kullanmak çok daha uygundur.

Komut	İkili	Onaltılı
STA HEX 006	0011 0000 0000 0110	(3006)h
LDA HEX 002	0010 0000 0000 0010	(2002)h

### 6.3 Karşılaştırmalı Program Örneği

Son olarak C dilinde yazılmış, iki sayının toplamını alan bir programın temel bilgisayar assembler dilinde yazılımını göstererek aralarındaki farkları belirtelim: C dilinde iki sayının toplamı aşağıda gösterilmiştir. Aynı sayıları tanımladığımız assembler dilinde toplarsak, aşağıdaki programı yazmamız gerekir.

Assembler komutları daha temel (alt seviye) komutlar olup, her bir operasyonu yapabilmek için hızlı bellek transferlerini tanımlamak gerekir. Yukarıdaki assembler programı -20 ile 40 sayısını toplamak için bölüm 5'te tanıtılmış olan AC (akümülatör) hızlı belleğini kullanmaktadır.

#### 6.4 Assembly Dili Detayları

Tüm programlama dilleri gibi, burada tanıtılmış olduğumuz assembler dili de belli kurallardan oluşur. Yazılan programların doğru çevrilmesi (assembling) için, kullanıcının bu kurallara tamamıyla uyması gerekir.

Bir Assembly programında her bir satıra 3 kolon bulunabilir;

**1-Etiket alanı:** Bu alan boş olabilir veya anahafıza üzerindeki bir adresin sembolik değerini tutar. Örnek: X, Y, Basla, Son,,,,:

**2-Kod Alanı:** Bu alanda bir makine dilindeki **kod** veya **pusedo code (direktif)** olabilir.  
**2a-Makine dilindeki kod** kullanılan mikroişlemci için tanımlanmış olan ve karşılığında bir mikroişlemci kodunun bulunduğu koddur.

Örnek ADD, LDA, STA

**2b-Pusedo kod** (direktif) ise kullanılan derleyicinin programı ne şekilde derlemesi gerektiğini belirten koddur ve bir makine dili karşılığı yoktur.

Örnek

ORG: programın başlangıç adresini gösterir,

END: Programın bittiğini gösterir

OPR: komutun operantını gösterir

I: indirek (dolaylı) adresleme modundaki komutu tanımlar

HEX: Onaltılık sayı sisteminde verilmiş bir sayıyı gösterir

DEC: Ondalık sayı sisteminde verilmiş bir sayıyı gösterir

**3-Açıklama Alanı:** Her assebly programında programın çalışmasını açıklayan ve / ile ayrılmış açıklama alanı bulunmalıdır.



## 6.5 Örnek Uygulamalar

**Örnek-1**  $z = (x-2*y)$  işlemini yapacak assembly kodu

Assembly Kodu	Makine Dili Kodu
ORG 100HEX / Başlangıç adresi direktif	<b>Ana bellek: İçerik</b>
LDA Y / Y değerini al	100: 2108
CIL / Sola kaydır . İki ile çarp	101: 7040
CMA /1 e göre tümleyenini al	102: 7200
INC / 1 arttırarak 2 ye göre tümleyenini al. Sayı negatif	103: 7020
ADD X / X sayısı ile topla	104: 1107
STA Z / sonucu Z adresine sakla	105: 3109
HLT / programı durdur	106: 7001
X: DEC 20 / x sayısı	107: 0014
Y: DEC 4 /y sayısı	108: 0004
Z: DEC 0 / sonucun saklanacağı adres	<b>109: 000C</b>
END / proramın sonu direktif	

**Örnek-2:** Ana bellekte arka arkaya saklanmış 5 sayıyı toplayan program.

C kodu	Assembly Kodu	Makine Dili Kodu	
main()	ORG HEX 100	Ana bellek	İçerik

<pre> { int sum, a[5], i, K=5; sum=0; for ( i=0; i&lt;=K-1; i++) sum= sum+a[i]; } </pre>	LDA ADR	100:	211B
	STA PTR	101:	311C
	LDA SAYI	102:	211D
	STA SAYICI	103:	311E
	CLA	104:	7800
	CEVRIM, ADD PTR I	105:	911C
	ISZ PTR	106:	611C
	ISZ SAYICI	107:	611E
	BUN CEVRİM	108:	4105
	STA TOPLAM	109:	311F
	HLT	11A:	7001
	ADR, HEX 250	11B:	0250
	PTR, HEX 0	11C:	0000
	SAYI, DEC -5	11D:	FFFB
	SAYICI, HEX 0	11E:	0000
	TOPLAM, DEC 0	11F:	<b>0000 ?</b>
	ORG HEX 250		
	DEC 1	250:	0001
	DEC 5	251:	0005
	DEC 4	252:	0004
	DEC 2	253:	0002
	DEC 8	254:	0008
	END		

**Örnek-3:** Ana bellekte x,y, ve z değerleri verilen sayılar üzerinde ((x AND y)') OR z mantıksal işlemini yapan program.

Assebly Kodu	Makine Dili Kodu	
	Ana bellek	İçerik
ORG HEX 100		
LDA x	100	2109
AND y // x & y	101:	010A
STA t	102:	311C
LDA z	103:	211B
CMA // z'	104:	7200
AND t // (x & y)' & t	105:	011C
CMA // ((x & y)' & t) = (x'+y') + t'	106:	7200

STA SONUC		107:	311D
HLT		108	7001
x,	HEX FFF0	109:	FFF0
y,	HEX F0F0	10A:	F0F0
z,	HEX 00FF	10B:	00FF
t,	HEX 0	10C:	?
SONUC,	HEX 0	10D:	?
END			

C dili	Assembly Dili ve Açıklama	Assembly List dosyası		Makine dili Hex gösterimi		Makine dili Binary gösterimi	
main ()  {int   x=-20, y=40; int z; z = x + y ; }	ORG DEC100       / Başlangıç adres bölgesini gösteren	Adres	İçerik	Adres	İçerik	Adres	İçerik
	<b>Direktif.</b>	100:	LDA[104]	064:	2068	0000 0110 0100:	0010 0000 0110 1000
	LDA X       / ACC ye X <b>etiketinin</b> değerini yükle	101:	ADD[105]	065:	1069	0000 0110 0101:	0001 0000 0110 1001
	ADD Y       /ACC değeri ile Y <b>etiketinin</b> içeriğini	102:	STA [106]	066:	306A	0000 0110 0110:	0011 0000 0111 0000
	topla	103:	HLT	067:	7001	0000 0110 0111:	0111 0000 0000 0001
	STA Z       / Sonucu Y <b>etiketinin</b> gösterdiği adrese	104:	-20	068:	FFEC	0000 0110 1000:	1111 1111 1110 1100
	sakla	105:	4	069:	0004	0000 0110 1001:	0000 0000 0000 0100
	HLT       / Programın çalışmasını durdur	106:		06A:	FFF0	0000 0110 1010:	1111 1111 1111 0000
	X, DEC -20   / X <b>etiketinin</b> ondalık değeri						
	Y, DEC 4     /Y <b>etiketinin</b> ondalık değeri						
Z, DEC 0     / Sonucun saklanacağı adresin <b>etiketi</b>							
END       / Programın sonunu gösteren <b>Direktif</b>							
Donanım tanımlama dili gösterimi		Saklayıcı ve Anabelleğin içerikleri			Makine dili Hex gösterimi	Açıklama	
R´T <sub>0</sub> :AR ← PC, SC ← SC+1 R´T <sub>1</sub> :IR← M[AR], PC ← PC+1, SC ← SC+1 R´T <sub>2</sub> :(D <sub>0</sub> ---D <sub>7</sub> )← Decode IR(12-14), AR ← IR(0-11), I ← IR(15) , SC ← SC+1 D <sub>2</sub> T <sub>4</sub> :DR ← M[AR] , SC ← SC+1 D <sub>2</sub> T <sub>5</sub> :AC ← DR, SC ← 0		PC=064 h, AR=064h, T=T <sub>1</sub> IR=2068h, PC=065h, T=T <sub>2</sub> AR=068h, I=0h, T=T <sub>3</sub> , T=T <sub>4</sub> DR=FFEC h, T=T <sub>5</sub> AC=FFEC h, T=T <sub>0</sub>			064:2068	PC nin başlangıcı ORG 100 (064h) ile olur	
R´T <sub>0</sub> :AR ← PC, SC ← SC+1 R´T <sub>1</sub> :IR← M[AR], PC ← PC+1, SC ← SC+1 R´T <sub>2</sub> :(D <sub>0</sub> ---D <sub>7</sub> )← Decode IR(12-14), AR ← IR(0-11), I ← IR(15) , SC ← SC+1 D <sub>1</sub> T <sub>4</sub> :DR ← M[AR] , SC ← SC+1		AR=065h, T=T <sub>1</sub> IR=1069h, PC=066h, T=T <sub>2</sub> AR=069h, I=0h, T=T <sub>3</sub> , T=T <sub>4</sub> DR=0004h, T=T <sub>5</sub> AC=FFF0h, E=0H, T=T <sub>0</sub>			065:1069		

D <sub>1</sub> T <sub>5</sub> : AC ← AC + DR, E ← Co, SC ← 0			
R' T <sub>0</sub> : AR ← PC, SC ← SC+1 R' T <sub>1</sub> : IR ← M[AR], PC ← PC+1, SC ← SC+1 R' T <sub>2</sub> : (D <sub>0</sub> ---D <sub>7</sub> ) ← Decode IR(12-14), AR ← IR(0-11), I ← IR(15), SC ← SC+1 D <sub>3</sub> T <sub>4</sub> : M[AR] ← AC, SC ← 0	AR=066h, T=T <sub>1</sub> IR=3070h, PC=067h, T=T <sub>2</sub> AR=070h, I=0h, T=T <sub>3</sub> , T=T <sub>4</sub> 0070h=FFF0h, T=T <sub>0</sub>	066:3070	
R' T <sub>0</sub> : AR ← PC, SC ← SC+1 R' T <sub>1</sub> : IR ← M[AR], PC ← PC+1, SC ← SC+1 R' T <sub>2</sub> : (D <sub>0</sub> ---D <sub>7</sub> ) ← Decode IR(12-14), AR ← IR(0-11), I ← IR(15), SC ← SC+1 D <sub>7</sub> I' T <sub>3</sub> B <sub>0</sub> : S ← 0, SC ← 0	AR=067h, T=T <sub>1</sub> IR=7001h, PC=068h, T=T <sub>2</sub> AR=001h, I=0h, T=T <sub>3</sub> , S=0, T=T <sub>0</sub>	067:7001	<b>Dikkat!</b> IR=7001h ise (D <sub>7</sub> =7h =0111b) I=0 B <sub>0</sub> =1 D <sub>7</sub> I' T <sub>3</sub> B <sub>0</sub> koşulu sağlandı