

Von Neumann Mimarisi

Sayısal Bilgisayarın Tarihsel Gelişim Süreci

- Babage'in analitik makinası (1833)
- Vakumlu lambanın bulunuşu (1910)
- İlk elektronik sayısal bilgisayar (1946)
- Transistörün bulunuşu (1947)
- İlk transistörlü sayısal bilgisayar (1960)
- Entegre devrenin bulunuşu (1963)
- İlk mikroişlemci (1970)
- 10.000 transistörlü entegre devreler (1981)

Saklı Program Kavramı

- ✓ Bilgisayardan istenilen işlerin gerçekleştirilebilmesi için gereken işlem dizisi bilgisayarın içinde saklanabilmektedir.
- ✓ İstenilen bir işi yerine getiren işlemlerin listesine program denir. Programlar bilgisayarın içinde saklanırlar.
- ✓ Program kavramının iki avantajı vardır; Birincisi işlemler için gereken zaman kullanıcıya bağlı değildir. İkincisi sadece veriler girilerek aynı işlemler tekrar tekrar yaptırılabilir.



Temel Yapı

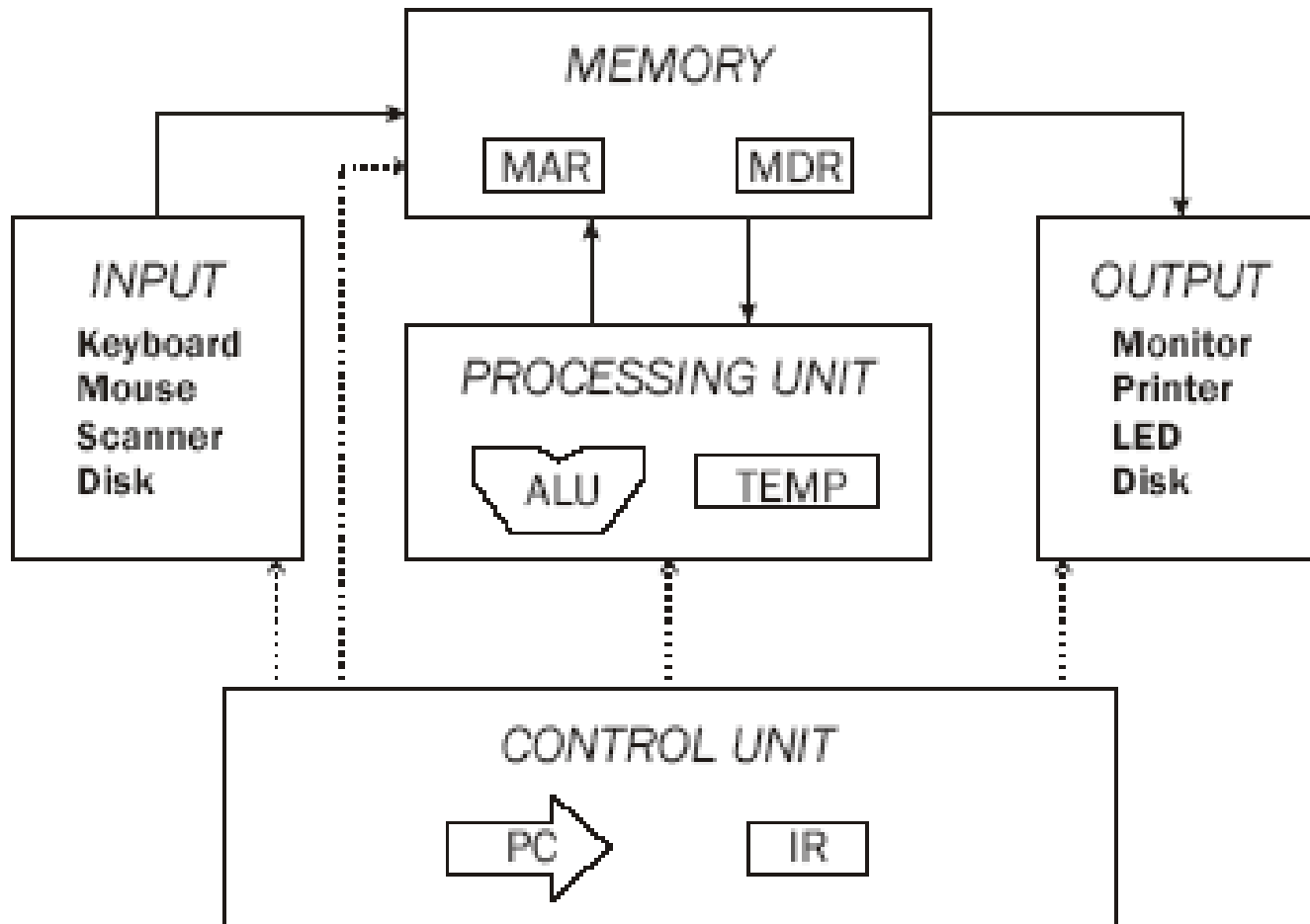
Sayısal bir bilgisayar 3 bileşenden oluşmaktadır.

- i.Hem programı oluşturan komutların listesini, hem de işlemeyi bekleyen veri değerlerini tutan bir bellek birimi
- ii.Her bir program komutunu yorumlayıp yürüten merkezi işlem birimi (MİB & CPU)
- iii.Bir veya daha çok giriş ve çıkış portu sağlayan giriş-çıkış (I/O) arabirim ünitesi



Mikroişlemci, ROM, RAM ve I/O birimlerinden oluşan adreslenebilir saklayıcılar topluluğudur.

Von Neumann Modeli



Bellek(Memory)

Bellek tümdevresinin kapasitesi arttıkça kullanılacak uç sayısı da **logaritmik** olarak artmaktadır.



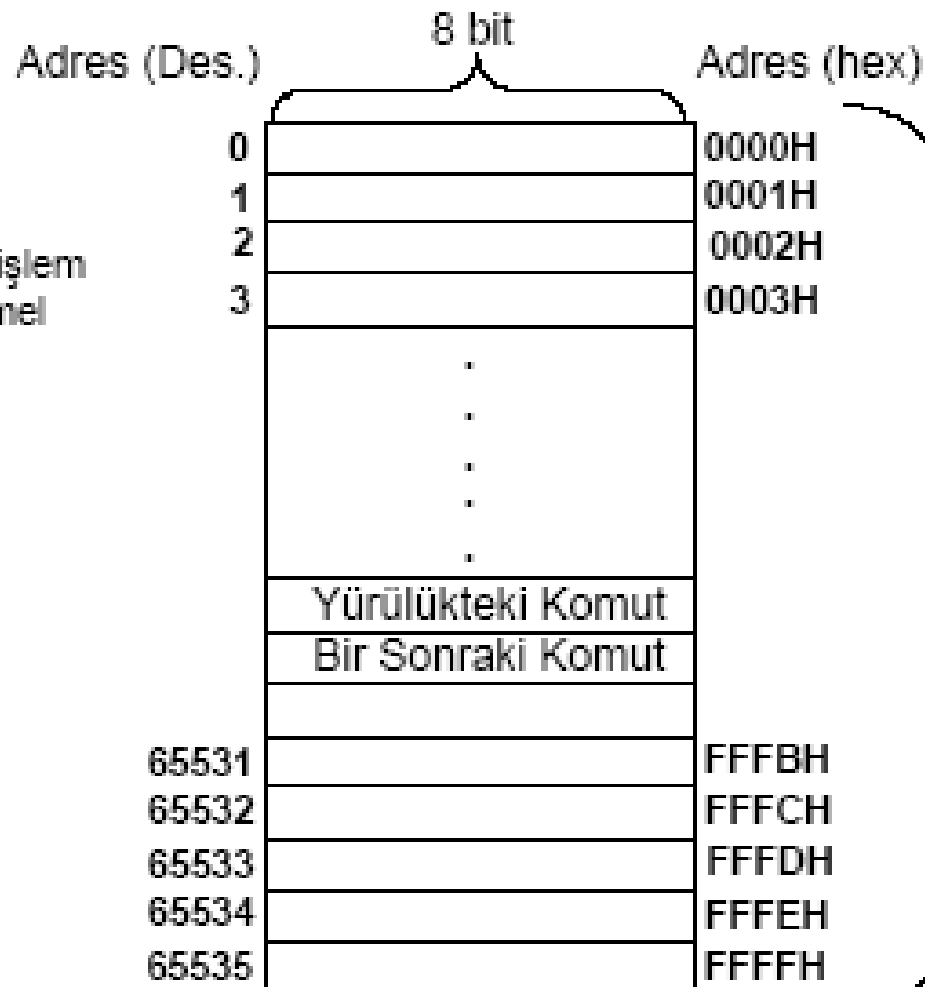
Temel bellek birimi gözlerdir.



Gözlerin bir araya gelmesiyle de bellek oluşur.



Bellek
üzerinde işlem
yapan temel
komutlar
LOAD ve
STORE

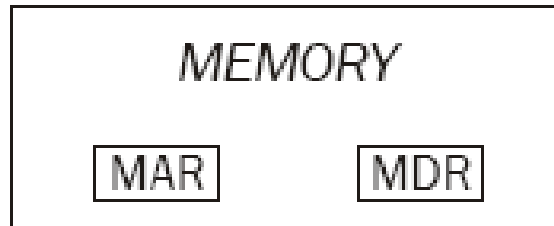


16 bit adres
hattına sahip
bir sistemin
bellek yapısı

Bellek Arayüzü(Memory Interface)



Mikroişlemci, bellek ile MAR (**Memory Address Register**) ve MDR (**Memory Data Register**) olarak bilenen iki yapı yardımıyla iletişim kurar.





Bellekteki X konumundan bir okuma işlemi gerçekleştirileceği zaman:




- Adres X ilk önce MAR kaydedicisine yerleştirilir.
- Belleğe bir okuma sinyali (RD) gönderilir.
- X konumunun içerdiği veri değeri MDR kaydedicisinde elde edilir.

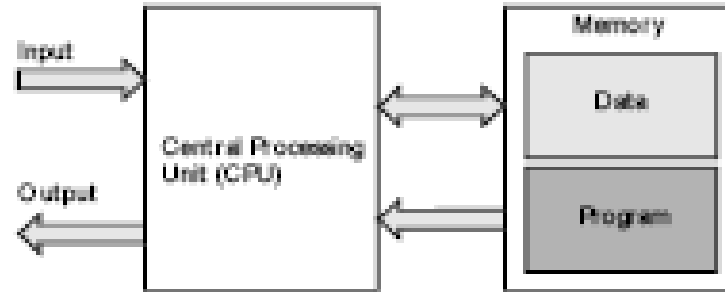


Bellekteki X konumuna T değerinin yazılması gerektiği durumda

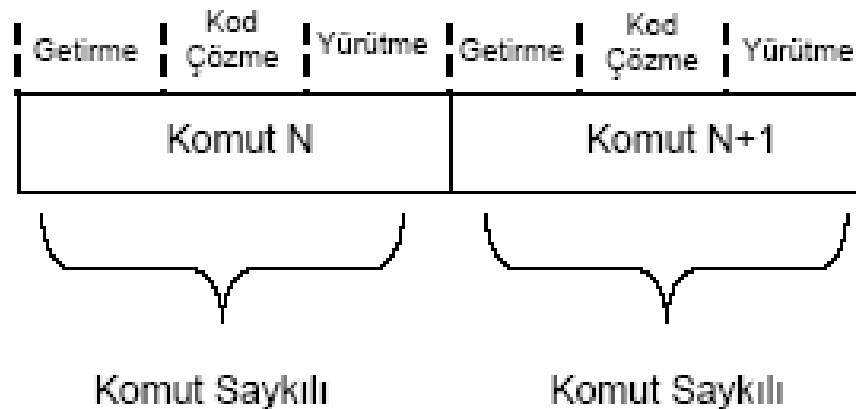
- T verisi önce, MDR kaydedicisinin içerisine yerleştirilir.
- X bellek adresi MAR kaydedicisine yerleştirilir.
- Ardından mikroişlemci denetim birimi tarafından üretilen yazma sinyali (WR) ile bu konuma T verisi yazılmış olur.

Merkezi İşlem Birimi (CPU)

-  Mikroişlemci, bellekten komutları okuyan, çalıştıran ve giriş-çıkış cihazlarını kontrol eden birimdir. Bilgisayarın beynidir denilebilir.
-  Bilgisayar içerisinde olan işlem akışlarını kontrol eder ve düzenler.
-  Bütün matematiksel ve mantıksal işlemleri gerçekleştirir.

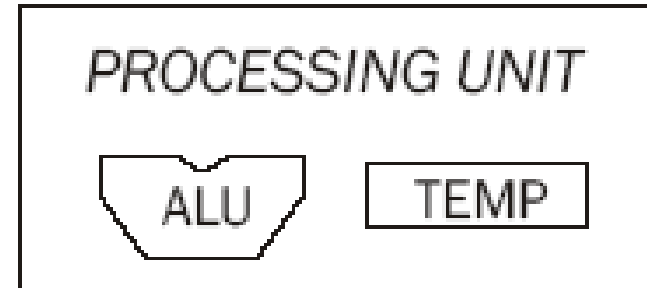


Mikroişlemci, bellekten komutları okuyan, yürüten ve giriş-çıkış cihazlarını kontrol eden birimdir.



✓ Arithmetical and Logic Unit (ALU)

Temel aritmetiksel (+,-,x,/) ve mantıksal (AND, OR, NOT, ...) işlemleri gerçekleştirir.



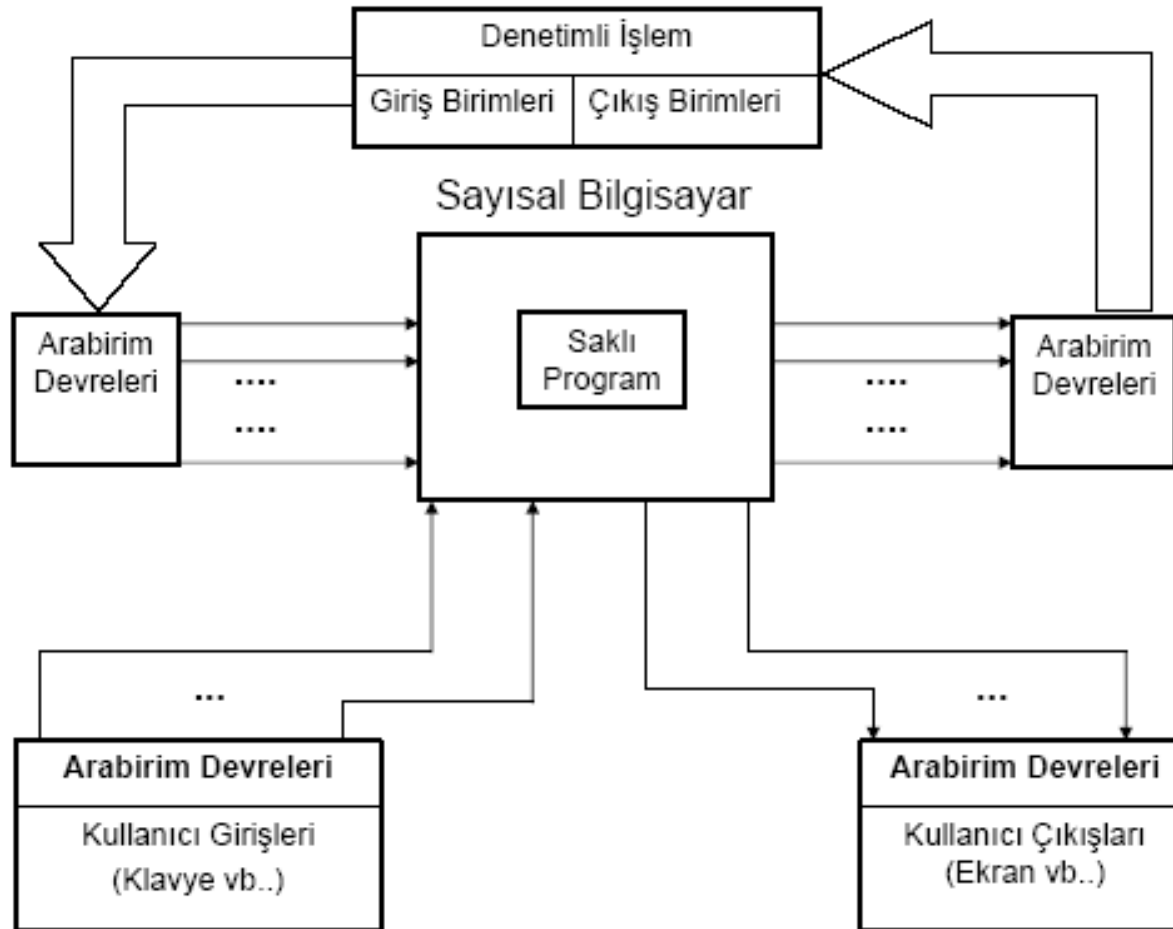
✓ Kaydediciler (Registers)

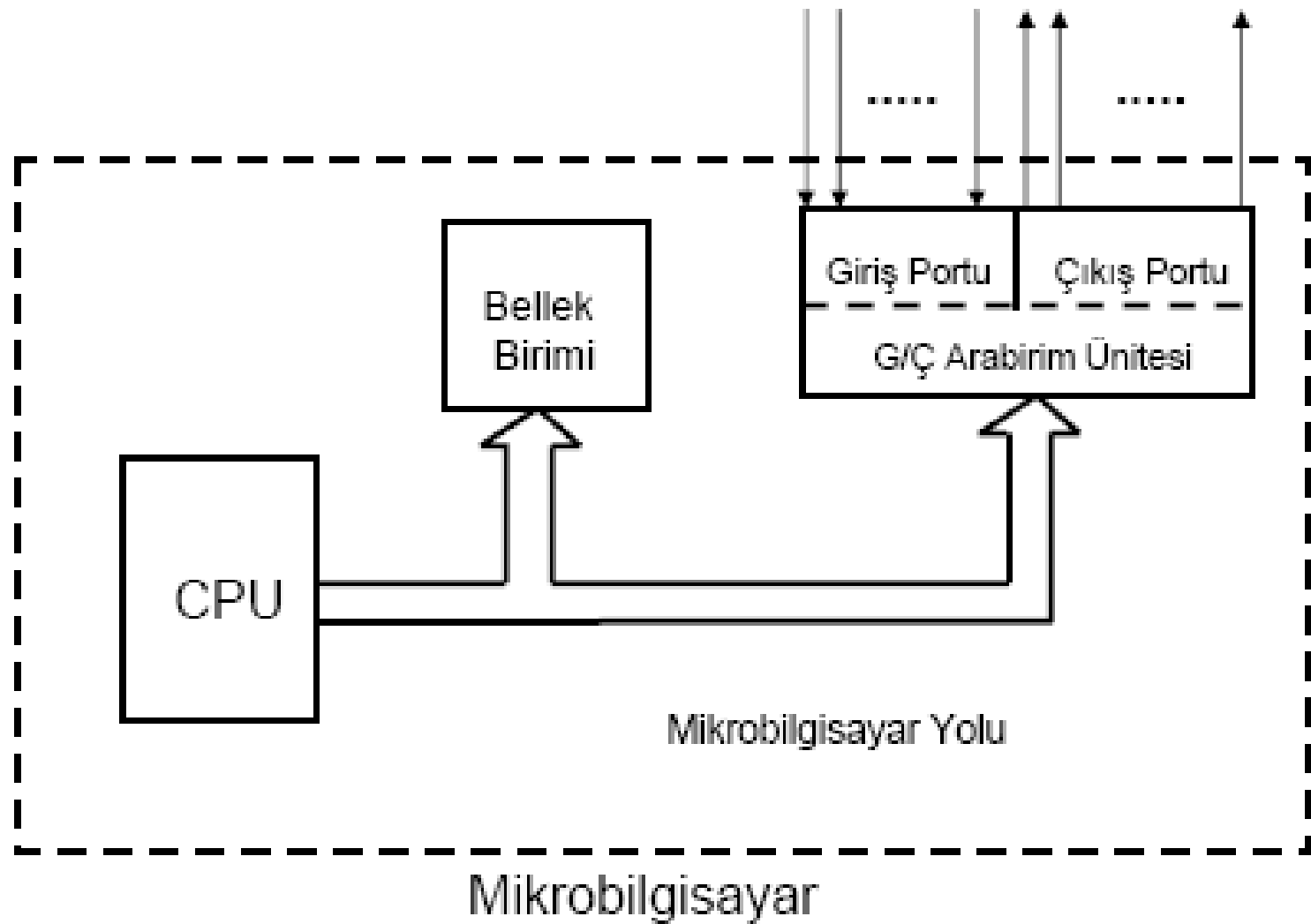
Üzerinde işlem yapılacak veri ve/veya komut parçacıklarının mikroişlemci içerisinde saklayan gözlerdir. Bellek gözleri ile benzerlik taşır.

✓ Kelime Uzunluğu (Word Size)

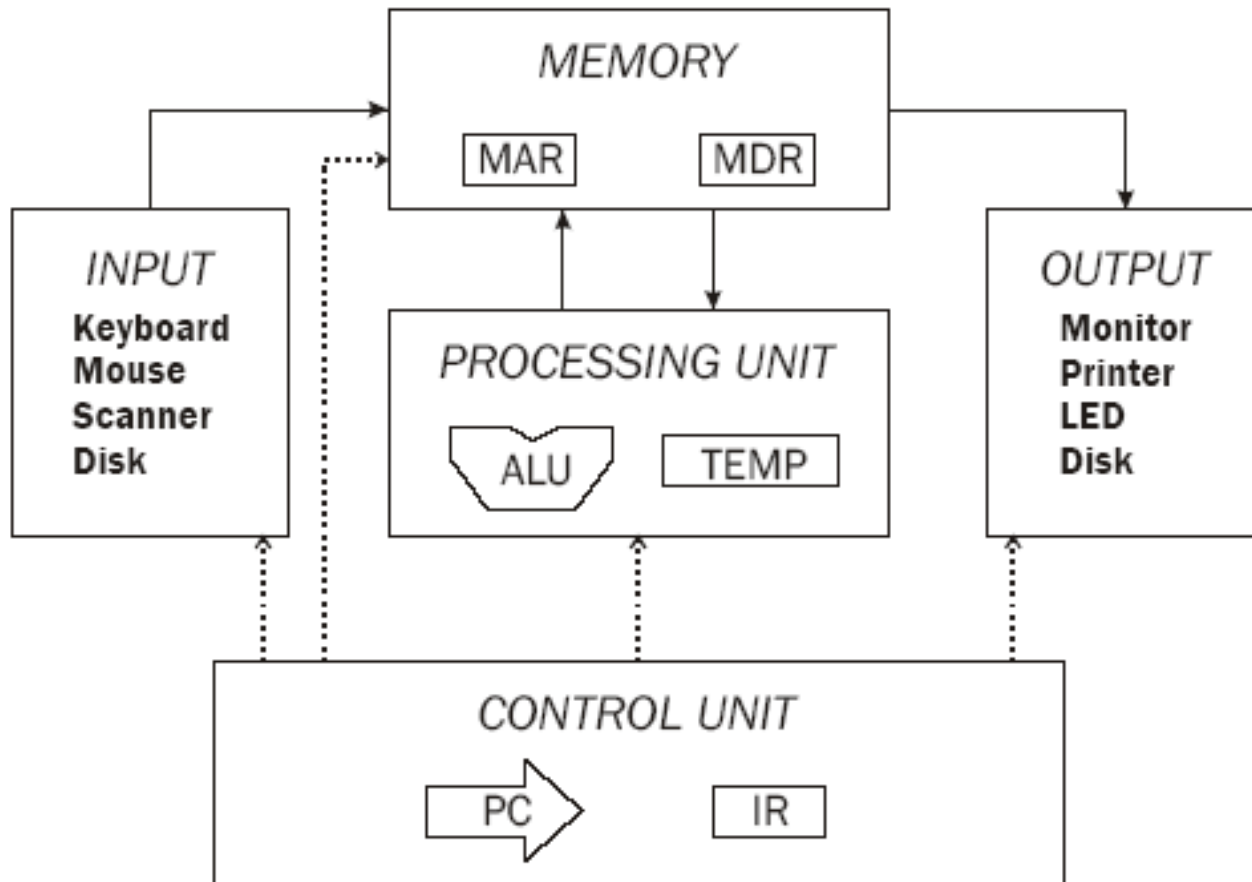
Paralel olarak işlenen veri bitlerinin sayısıdır. Kelime, işlemcideki genel amaçlı kaydedicilerin büyüklüğü ve aynı zamanda her bir bellek alanı kapasitesidir.

Giriş/Çıkış (Input & Output) Arabirimleri





Von Neumann Modeli




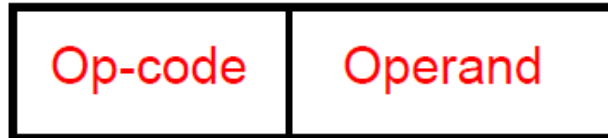
Kontrol Birimi (Control Unit)



- ✓ Komutların yürütümünü gerçekleştirir. Sistem üzerindeki senkronizasyonu sağlar. Sistemdeki diğer birimlere ne yapmaları gerektiğini söyler.
- ✓ Komut Kaydedicisi (Instruction Register) : O an yürürlükte olan komutun tutulduğu kaydedicidir.
- ✓ Program Sayıcısı (Program Counter) : Bellekte yürütülecek bir sonraki komutun adresini tutar.

Komut


 Komut, yapılan bir işin en temel parçasıdır.



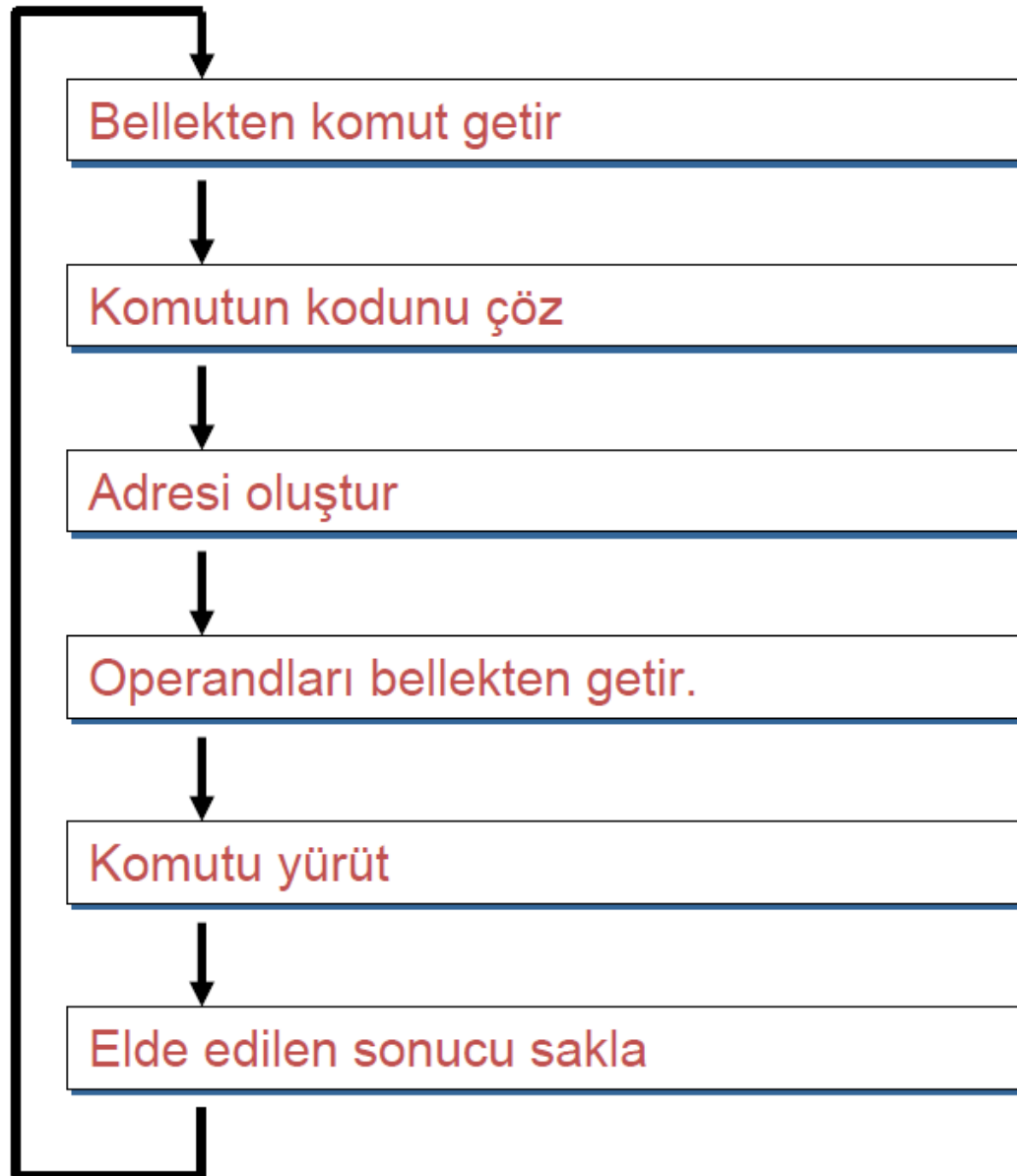
Op-code (İşlem Kodu): CPU'ya hangi işlemin gerçekleştirileceğini söyler.

Operand (İşlenen): Op-code tarafından kullanıcak verinin adresini belirtir.

Mikroişlemci içindeki komutlar birden fazla operanda sahip olabileceği gibi hiç bir operandı olmayan komutlarda mevcuttur.

 Bilgisayarın komutları ve formatları, komut seti mimarisi olarak bilinir (**Instruction Set Architecture-ISA**)

Komut İşleme



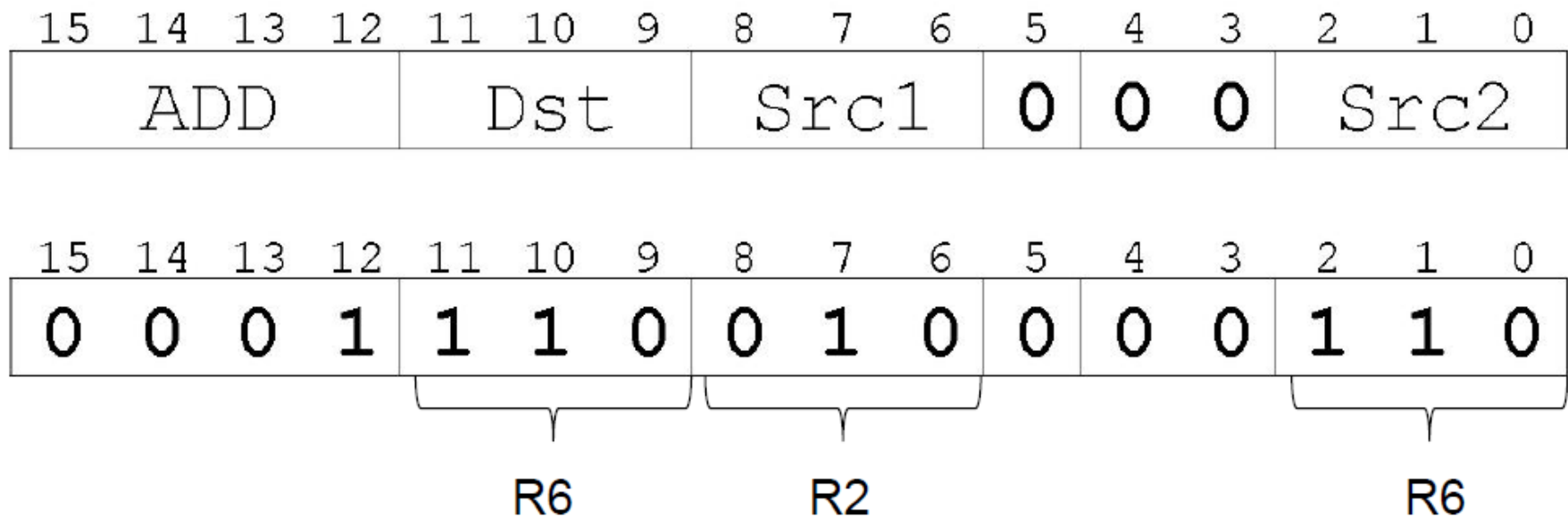
Örn: T mikroişlemcisinde ADD komutunun yürütülmesi



T mikroişlemcisi 16 bitlik komut setine sahiptir. 16 bitlik komut setinin 4 biti işlem kodu (op-code) için ayrılmıştır. [12-15]



T mikroişlemcisinin R0,...,R7 olmak üzere 8 adet kaydedicisi vardır.



Bu komut, R2'nin içeriğini R6'nın içeriğine ekler ve sonucu R6'da saklar.

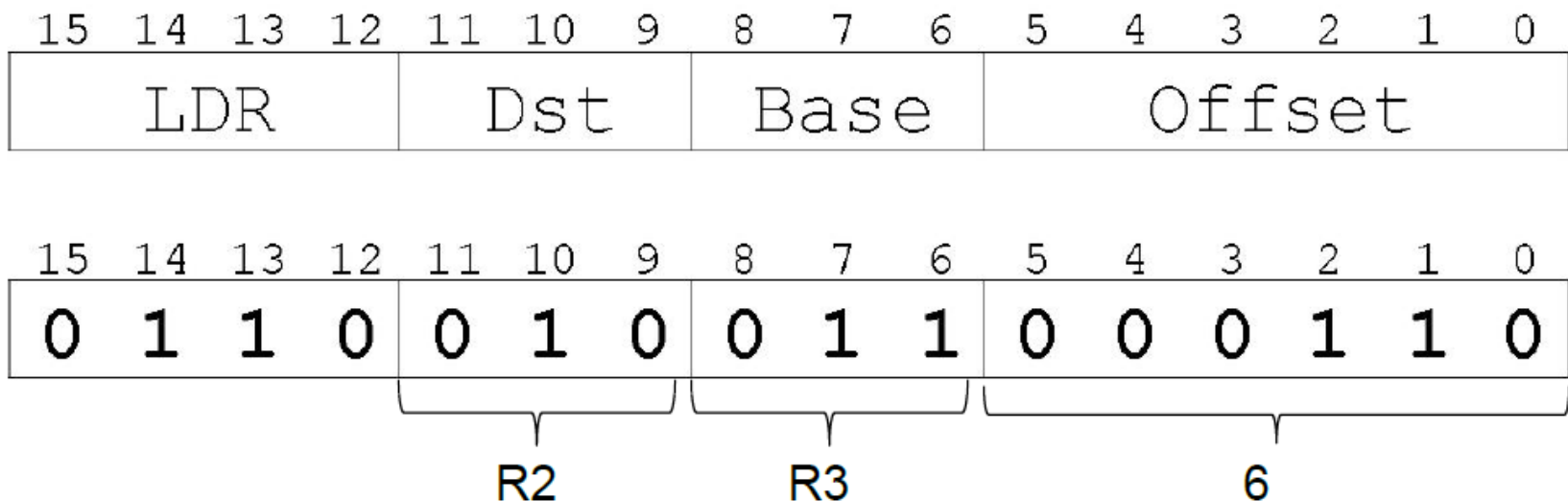
Örn: T mikroişlemcisinde LDR komutunun yürütülmesi



Veriyi bellekten okur ve okunan değeri belirtilen kaydedicilerden birisinde saklar.

$$\text{Offset Address} + \text{Base Address} = \text{Memory Address}$$

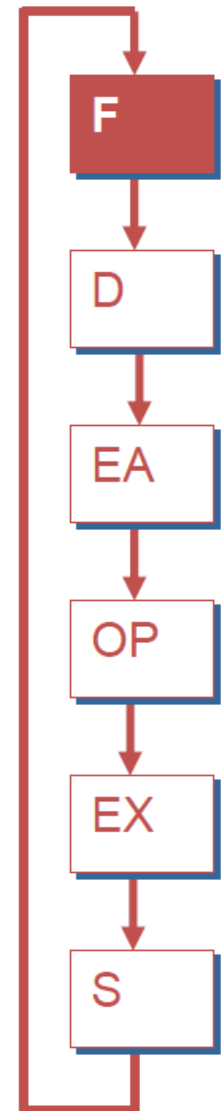
Memory Address ile belirtilen adresteki değeri "0 1 0" ile belirtilen hedef kaydedicide saklar.



Bellek adresini oluşturmak için 6 değerini R3 kaydedicisinin içeriğine ekler. Ardından oluşturduğu bellek adresinin içeriğini R2 kaydedicisinde saklar.

Komut İşleme: Getir(Fetch)

- ✓ PC'nin içeriğini bellek adres kaydedicisine yükle (MAR)
- ✓ Bellekten bir okuma işlemi gerçekleştir.
- ✓ Bellek veri kaydedicisinin (MDR) içeriğini oku ve bunu daha sonra komut kaydedicisine (IR) yerleştir.
- ✓ PC'nin içeriğini bir sonraki komutu gösterecek şekilde 1 arttır.



Komut İşleme: Kod Çözme(Decode)



Komutun işlem parçasını (op-code) tanımla.

- T mikroişlemcisinde bu komutun en yüksek değerlikli ilk 4 bitidir.

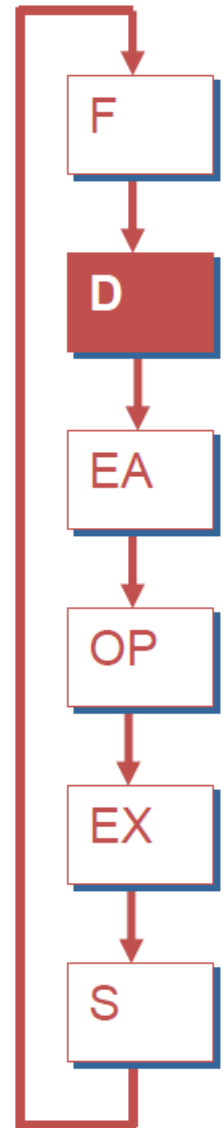


Mikroişlemci komutun op-code kısmına bakarak operand kısmı için kaç byte getireceğini bilir.

Örn:

LDR komutunun son 6 biti offset adresini gösterir.

ADD komutunun son 3 biti işlem parçasının ikinci kısmını (operand 2'yi gösterir)



Komut İşleme:

Adresin Hesaplanması (**E**valuate **A**ddress)



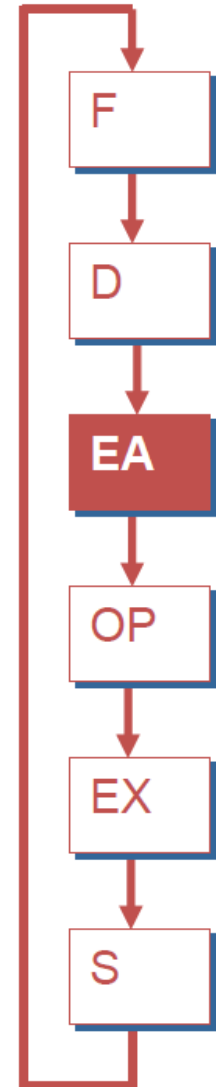
Bellek erişimine ihtiyaç duyan bir komut, erişim için gerekli olan adresi hesaplar.

Örn:

Offset Address + Base Address = Memory Address

Offset Address + PC = Memory Address

Offset Address + 0 = Memory Address



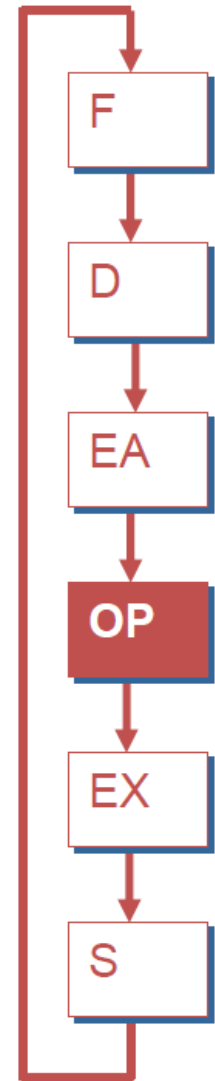
Komut İşleme:İşlenen Getir(Fetch **O**perand)



Komutun üzerinde işlem yapacağı parçaları getirir.

Örn:

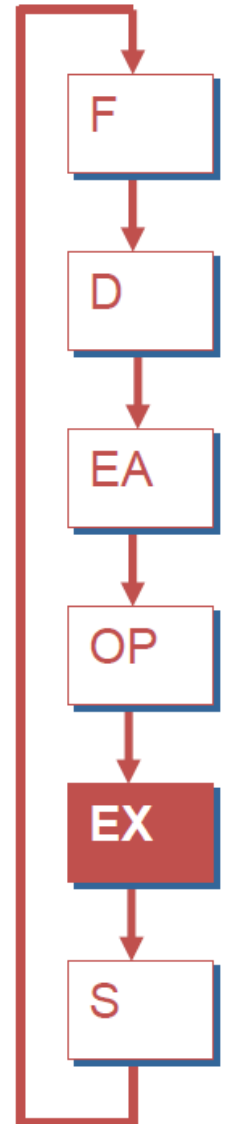
- Bellekten verinin yüklenmesi
- Kaydediciden bir değerin okunması



Komut İşleme:Yürütme (Execution)



Mikroişlemci önceden okumuş olduğu op-code ve operandları kullanarak ilgili işlemi yerine getirir.



Komut İşleme: Sakla (Store)

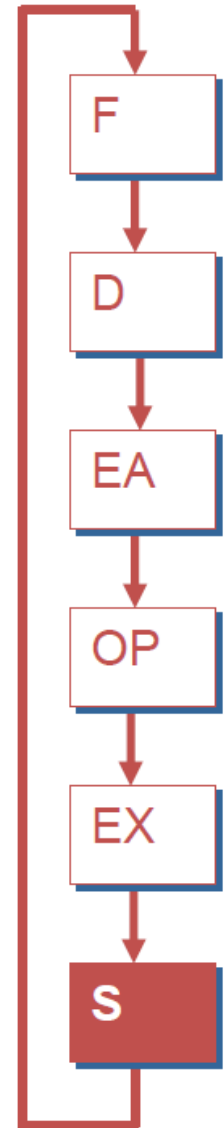


Komut sonucunda, elde edilen değer saklanır. Bu bir bellek konumu olabileceği gibi, bir kaydedici veya bir çıkış portu da olabilir.




Örn:

Bellekteki bir konuma sonucun saklanacağı durumda

- Önce ilgili konumun adresi MAR kaydedicisine yerleştirilir
- Kaydedilecek veri MDR kaydedicisine yerleştirilir.
- Ardından belleğe yazma işlemi gerçekleştirilir.



Komut İşlenme Sırasını Değiştirme

-  Buraya kadar belirtilen işlemlerde bellekteki komutlar sırasıyla işlenmekteydi. Bu durumda PC'in içeriği sırasıyla birer birer artmaktaydı.
-  Bazı durumlarda bu tür kullanımın dışına çıkılarak program akışının yönünü değiştirmek mümkündür. **Örn:** Döngüler(loop), Şartlı yapılar (if-then) ve alt yordamlar (sub-routines) gibi.
-  Bu işlemi gerçekleştirmek için bazı komutlara gereksinim duyulur. Bunlar sıçırma(jump) ve şartlı dallanma(branch) komutlarıdır.

Komut Türleri



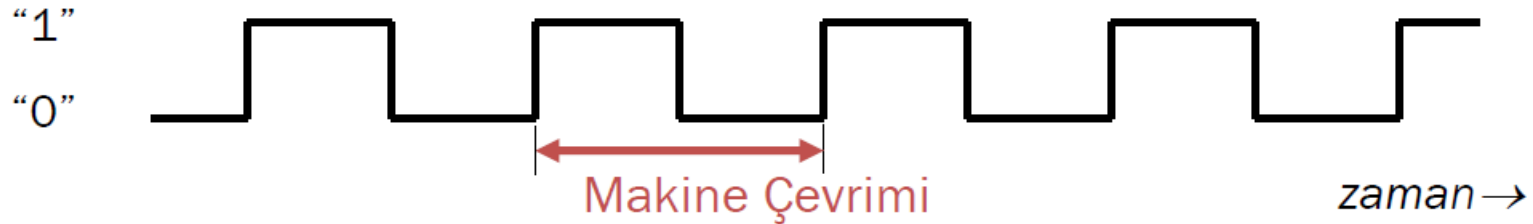
Mikroişlemci tarafından yürütülen komut türleri genel olarak aşağıdaki sınıflara ayrılabilir.

- Veri Taşıma Komutları
- Veri İşleme Komutları
- Program Kontrol Komutları
- Giriş / Çıkış Komutları

Sistem Saati (System Clock)

✓ Mikroişlemcide kontrol biriminin çalışmasını sağlayan sistem saatidir.

Saat darbesinin her bir kenarında kontrol birimi bir sonraki makine çevrimine girer. Bu çevrim yeni bir komut olabileceği gibi yürürlükteki komutun farklı bir evresi de olabilir.



Sistem saati, kristal bir osilatör devre yardımıyla yürütülür. Sistem saati düzenli olarak "1" ve "0" lar üretir.

Mhz ve MIPS

MIPS = millions of instructions per second

MHz = millions of clock cycles per second



Bu iki kavram birbirine benzer olmasına karşı farklıdırlar. **Neden?**

MegaHertz (MHz) her saniye başına saat çevrimlerinin sayısıdır. Örneğin bir işlemci basit bir komutu 12 saat çevriminde gerçekleştirsin. Bu 12 saat çevrimi 1 tane makine çevrimi olarak adlandırılır. Daha yeni bir işlemcimiz olsaydı ve her bir makine çevriminde sadece 4 saat çevrimine gerek duysaydı, bu işlemci ile aynı saat frekansında 3 kat daha hızlı (3 kat daha fazla komut işleyerek) iş yapmak mümkün olacaktı.

Saniye başına milyon komut sayısı olan MIPS (millions of instructions per second) saat frekansından farklı olarak kaç tane geçerli komutun işlendiğinin ölçütüdür.

(devamı diğer sayfada →)

Mhz ve MIPS

MIPS = millions of instructions per second

MHz = millions of clock cycles per second



Bu iki kavram birbirine benzer olmasına karşı farklıdırlar. **Neden?**

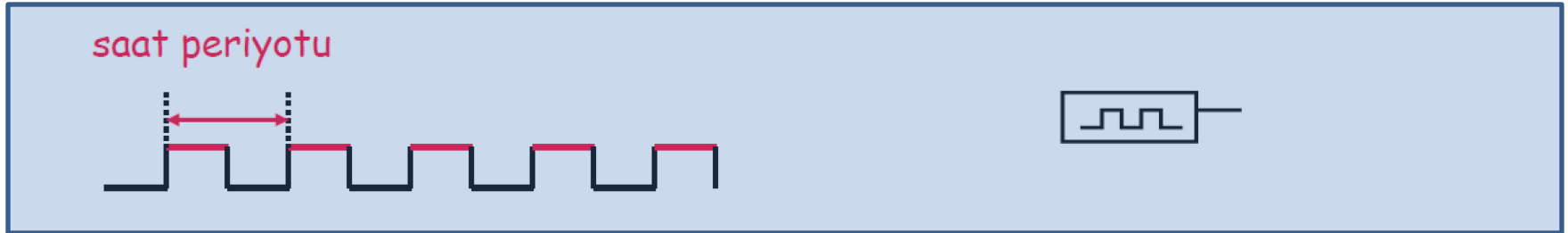
Eski model bir işlemcimiz olsun, bu işlemci her bir makine çevrimi başına 12 saat çevrimi ile çalışsın ve saat hızı 24 Mhz olsun. Buna göre bu makine 2 MIPS ($24 \text{ Mhz} / 12 \text{ cycle}$) işleyebilmektedir. Fakat daha yeni bir işlemcimiz olsaydı ve her bir makine çevriminde 4 saat çevriminde işleyebilen çekirdeklere sahip olsaydı, komutları 6 MIPS ($24 \text{ Mhz} / 4 \text{ cycle}$) ile işleyebilirdi, bunun anlamı aynı saat ölçüsüne göre 3 kat daha hızlı işlem yapmaktır.

Benzer tipte CPU'ları karşılaştıracaksak, MIPS ölçüsü, MHz gibi hız ölçüsünden daha anlamlı bir karşılaştırma imkanı sunar. Sonuçta hız işlemciden işlemciye değişkenlik gösterebilmektedir.

Saat ve Senkronizasyon

HATIRLAYINIZ:

- **Saat** çıkışı sürekli olarak belirli bir periyot ile 0 ve 1 arasında değişen özel bir devre elemanıdır.



- Saat'in 1'den 0'a değişmesi ile başlayan ve tekrar 1 oluncaya kadar geçen süreye **saat periyodu**, veya **saat devir süresi** denilir.
- **Saat frekansı** saat periyotunun tersidir. Birimi ise **hertz** dir.

- Saatler genellikle devrelerin senkronizasyonu için kullanılır. Devrelerde belli işlemlerin başlaması için tetikleme amaçlı kullanılırlar.
- Birden fazla devre aynı saati kullanırsa senkronizasyon sağlanmış olur. Bu, insanların senkronizasyon için saat kullanmalarına benzer bir durumdur.

Saat ve Senkronizasyon

- Saatler büyük olarak bilgisayar mimarisinde kullanılmaktadır.
- Tüm işlemciler bir iç saat ile çalışmaktadır.
 - Modern işlemciler (chip ler) 3.2 GHz'e kadar uzanan frekanslarda çalışmaktadır.
 - Bu da cycle time ı 0.31 ns kadar küçültmektedir!
- Dikkat... Daha yüksek frekans her zaman için daha hızlı makineye karşılık gelmez!
 - Her bir saat periyotunda ne kadar iş yapılabileceğine bakmak gerekir.
 - Ne kadar eleman 0.31 ns gibi sürede iş yapabilir?

Bu iç saatin kullanıldığı, bu sayede yazmaç (register) ve bellek (memory) birimlerinde veri saklama (depolama) işlemini yapabilmemizi sağlayan devreler olarak **Flip-Flop**'ları inceleyelim...

Flip-Flop'lar

- İki durumlu devrelerdir. İki kalıcı durumu olan ve çıktısı son durumu ile giriş değerlerine göre kalıcı durumdan birini alan öğedir. Bunlar sayaç (counter) devrelerinde, yazmaç (register) olarak ve bellek (memory) devrelerinde kullanılırlar.
- Flip-Flop'lar yapısında lojik (mantıksal) kapılar olan, yani lojik kapılar ile gerçekleştirilmiş özel elemanlardır. Girişlerin değişimine bağlı olarak çıkış değeri değişir. Flip-Flop'ların bu anlık değişimine **tetikleme** (triggering) denilir. Flip-Flop'lar ardışıl (sequential) devrelerde kullanılır, bir zamanlama (saat) durumu vardır.

Flip-Flop'lar

- Lojik devreler, Kombinasyonel (Combinational) ve Ardışıl (Sequential) olmak üzere iki bölümde incelenebilir.
- **Kombinasyonel devrelerde**, herhangi bir andaki çıkış, sadece o andaki girişler tarafından belirlenir. Önceki çıkış değerlerinin sonraki çıkışa hiç bir etkisi söz konusu değildir.
- **Ardışıl devrelerde**, ise bir önceki çıkış mevcut girişlerle birlikte sonraki çıkışı tayin eder. Başka bir deyişle ardışıl devrelerin bellek özelliği vardır. Yani çıkışları aklında tutar.

Flip-Flop'lar

Flip flopların genel özellikleri şunlardır: Her birinde clock (saat) girişi bulunmaktadır. Bu girişe kare dalga şeklindeki tetikleme sinyali bağlanır ve flip-flop bu sinyal ile çıkışlarını değiştirir. Flip-floplarda ise çıkışların değişmesi için girişlerin değişmesi yetmez. Bu değişim emrini tetikleme sinyali verir.

Flip-flopun vereceği çıkış girişlere bağlı olmakla birlikte, aynı zamanda bir önceki çıkışa da bağlıdır. Yani bir geri besleme söz konusudur. Bir önceki çıkış, sanki bir sonraki çıkışın girişi gibi düşünülür. Flip-floplar;

- Girişlerine uygulanan sinyal değişmediği müddetçe çıkış durumunu korur.
- Flip-floplar 1 “bit”lik bilgiyi saklayabilir.
- Giriş sinyallerine göre çıkış ya lojik “0” ya da lojik “1” olur.
- Her bir flip flobun Q ve \bar{Q} olmak üzere 2 çıkışı vardır. Q çıkışı “1” ise \bar{Q} “0” , Q çıkışı “0” ise \bar{Q} “1” olmaktadır. Uygulamada hangi çıkış işimize yarayacaksa o kullanılır. Esas çıkış Q çıkışıdır. Eğer Q çıkışının değilini kullanmak gerekirse ayrıca bir “Değil” kapısı kullanmaya gerek yoktur.
- Flip floplar ardışıl devrelerin temel elemanıdır.

Flip-Flop'lar

Dört temel tip Flip-Flop vardır:

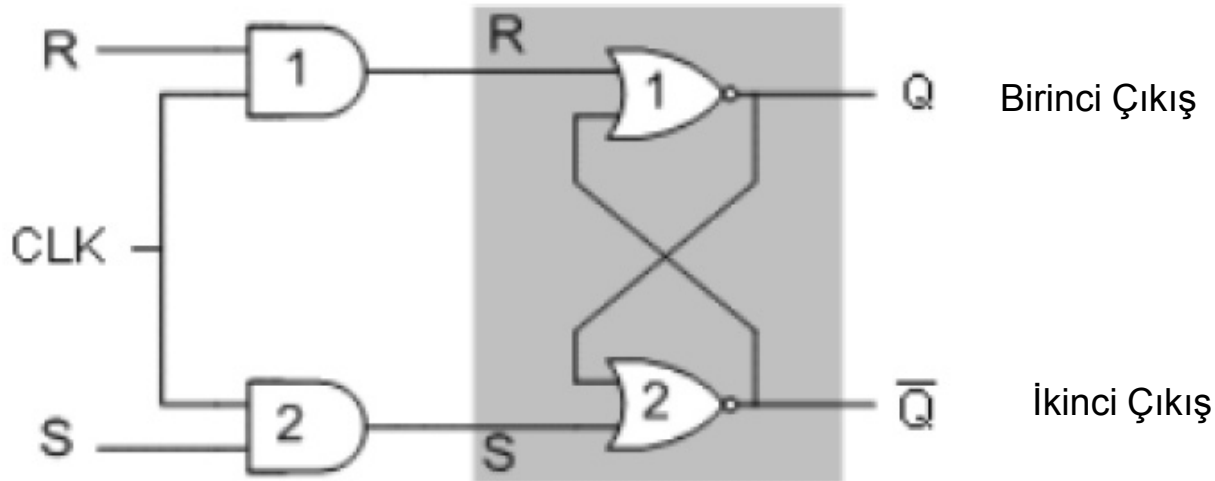
- 1) RS Flip-Flop'lar
- 2) JK Flip-Flop'lar
- 3) T Flip-Flop'lar
- 4) D flip-Flop'lar

Tetikleme palsi (elektriksel darbe, yani sonlu süreli sabit genlikli değişim)

0 volt ila 5 volt arasında değişen bir kare dalgadır. Yani lojik olarak "0" veya lojik olarak "1" olan bir işarettir.

Flip-Flop'lara örnek olarak **RS Flip-Flop** verelim:

R = Reset ve **S** = Set , **CLK** = Clock (saat) anlamına gelir.



İşlemci mimarisi

WORD:

WORD, işlemcinin bilgiyi taşıma birimi için belirlenmiş bir büyüklüktür. Yazmaç'ların (işlemcinin kullandığı kaydediciler) belirlenmiş genişliğini ifade eder. Veriyolu ve bellek adreslerine uygun oluşturulurlar.

- 16 bit işlemci için **WORD** = 2 Byte = $2 * 8 \text{ bit} = 16 \text{ bit}$
- 32 bit işlemci için (double)WORD = **DWORD** = 4 Byte = $4 * 8 \text{ bit} = 32 \text{ bit}$ (Şu an masaüstü en düşük konfigürasyondaki modern işlemciler)
- 64 bit işlemci için (quad)WORD = **QWORD** = 8 Byte = $8 * 8 \text{ bit} = 64 \text{ bit}$ (Şu an masaüstü en yüksek konfigürasyondaki modern işlemciler)

Geriye uyumluluk (Backward compatibility) :

- Doğal olarak WORD büyüklüğünü tüm mimariler desteklemektedir. Bu en doğalı 16 bit'lik olanıdır. Bunun anlamı, 64 bit veya 32 bit bir makine bile 16 bit'lik WORD'u destekler. Günümüzde en düşük konfigürasyonlu makine 32 bit mimarili işlemciye sahip olduğu için (double)WORD yani DWORD doğrudan kullanılmaktadır. Programlama dillerinde bu ayrım donanımı ilgilendirdiği için çok fazla dikkat edilir. Bununla beraber 64 bit mimarili makineler 32 bit ve 16 bit olan WORD'u desteklemektedir. Bunu geriye uyumluluk olarak ifade edebiliriz.

Karakter setleri ve iletişim

- **ASCII (Amarican Standard Code for Information Interchange) :**

Karakterler (eğer birden çok ise string olarak ifade edilir) bit dizisi şeklinde kodlanırlar (encoding). Buna dair en bilinen format ASCII formatıdır.

Modern mimarili makinelerde 127 karakteri standart ve uzatılmış versiyonu ile 256 karaktere kadar kullanılır. Karakterlerin herbiri decimal, octal veya hexadecimal olarak tablo ile verilir. Sonra bunlar bellekte uygun bit dizisi olarak depolanır. 7 bit'lik bir biçimde gösterilir. (MSB'deki bit sıfır değeri ile gösterilebilir).

- Örneğin Büyük A harfi, Büyük B harfi ... Büyük Z harfi:

<u>Char</u>	<u>Decimal</u>	<u>Hexadecimal</u>	<u>Binary</u>
A	65	41	01000001
B	66	42	01000010
..... Diğer harfler			
Z	90	5A	01011010

Bilgisayar Mimarileri

- Bilgisayar mimarileri için bir çok sınıflandırma bulunmaktadır. Bunlardan en bilineni **Flynn Sınıflandırması** (Flynn Taxonomy) olarak verilmektedir.
 - 1966 yılında Flynn tarafından ifade edilmiştir.
 - Dört adet sınıf içerir.
 - Kombinasyonlu (karışıklı) kullanıma da izin verir.

Bu dört sınıf:

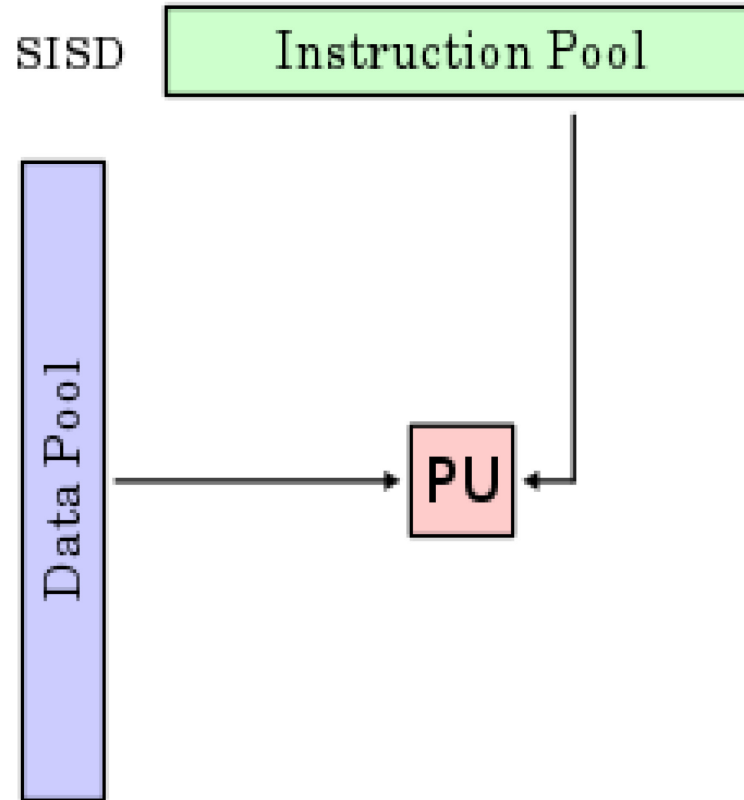
- 1) **SISD** (Single Instruction Single Data = Tek Komut Tek Veri)
- 2) **SIMD** (Single Instruction Multiple Data = Tek Komut Çoklu Veri)
- 3) **MISD** (Multiple Instruction Single Data = Çoklu Komut Tek Veri)
- 4) **MIMD** (Multiple Instruction Multiple Data = Çoklu Komut Çoklu Veri)

SISD

Single Instruction Single Data (Tek Komut Tek Veri)

- Seri bir bilgisayar hem komut yürütürken hem de veri akışlarını işlerken eşzamanlı (paralel) iş yapmaz. Buna geleneksel tek işlemcili PC'ler örnektir (Bunları tek çekirdek olarak görebiliriz).
- Buna göre tüm işlemcilerin gerek tek ortak bir yerden komutları aldığı gerekse tek bir ortak kaynaktan verileri edindikleri bilgisayar mimarisidir.

SISD



Anlamları:

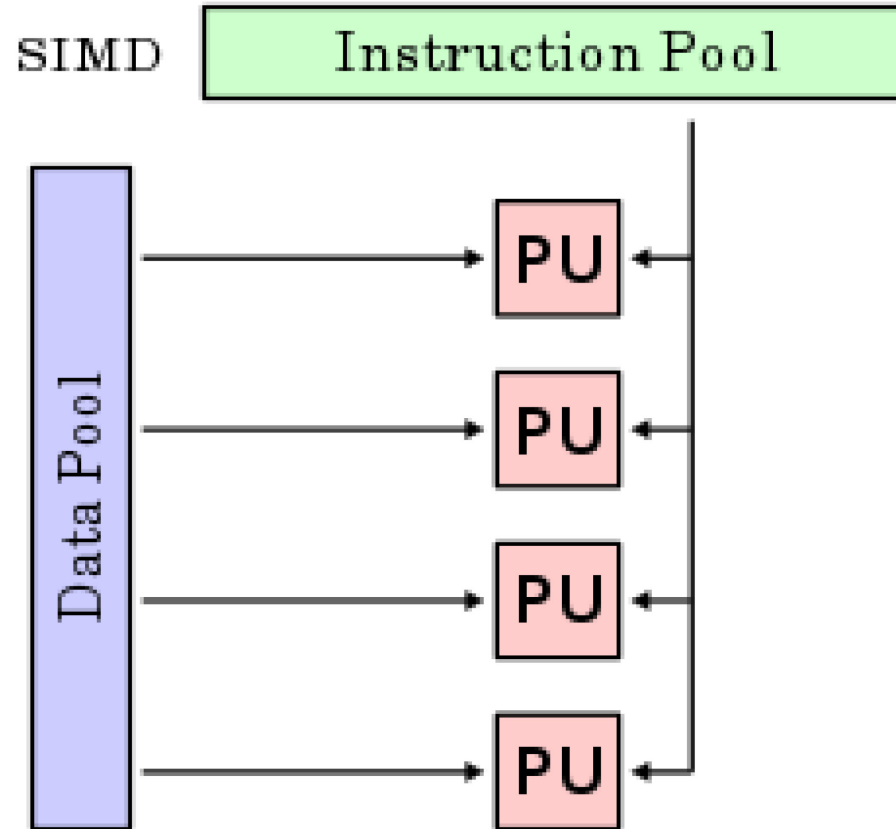
Data pool = Veri havuzu, **Instruction Pool** = Komut Havuzu, **PU** = İşlem birimi (İşlemci, Processing Unit).

SIMD

Single Instruction Multiple Data (Tek Komut Çoklu Veri)

- Tüm işlemcilerin tek bir ortak yerden komutları aldığı, verileri ise bağımsız bir çok kaynaktan edindikleri bilgisayar mimarisidir.
- Birden çok veri akışına karşılık tek komut akışı ile işlem yapılır. Buna örnek olarak Grafik İşlem Birimi (yani **ekran kartı işlemcisi**) (Graphic Processing Unit, **GPU**) örnek olarak verilebilir.

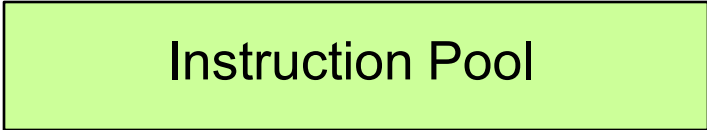
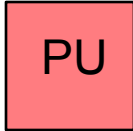
SIMD



Anlamları:

Data pool = Veri havuzu, **Instruction Pool** = Komut Havuzu, **PU** = İşlem birimi (İşlemci, Processing Unit).

SIMD için Ön uç işlemci (Front-end processor)

- Bir işlemci ana programı başlatır buna **ön uç işlemci** (front end processor) denilir.
 - Program kodunu diğer işlemcilere bu ön uç işlemcisi gönderir.
- Şekillerde yeşil renkte gösterilen bu ön uç işlemcidir. 
- Bu ön uç işlemci, bireysel işlem birimlerine (PU'lara) kodu gönderince bunlarda kendilerine atanan kodu  çalıştırmaya başlarlar. Şekillerdeki kırmızı renkte gösterilen işlemcilerdir.

SIMD Program Kodu'nun çalıştırılması

- Her bir işlemci aynı kodu çalıştırır.
- İşlemci numarası'na (CPU ID) göre kodda değişiklikler olabilir.
- Örnek: Verilen A dizisindeki elemanların toplamı

for each P_i ($i = 0$ to 9) // Ön uç işlemci tarafından çalıştırılır

$T_i = 0$

for $j = 0$ to 9 // Her bir P_i işlemcisi kendine düşen alanındaki toplamı hesaplar

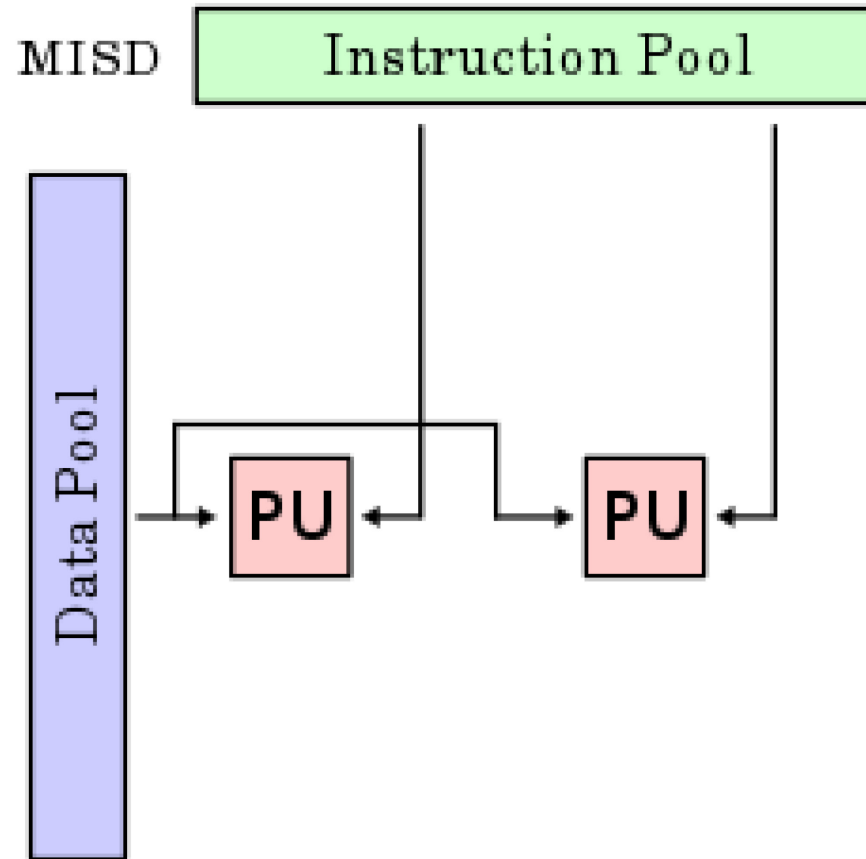
$T_i = T_i + A[i * 10 + j]$

MISD

Multiple Instruction Single Data (Çoklu Komut Tek Veri)

- Tek veri akışı üzerinde çoklu komutlar işletilmesidir. Hata toleransına yönelik uygulamalar için oluşturulmuştur.
- Homojen bir işlemciler ağının, veriyi işlemesi ve her biri düğüm elemanı olan işlemcilerle bir veriyi aktarması yoluyla işlem yapılır.
- Uzay mekiğini kontrol eden bir bilgisayar buna örnek olarak verilebilir.
- Daha az popüler olan bir yaklaşımdır.

MISD



Anlamları:

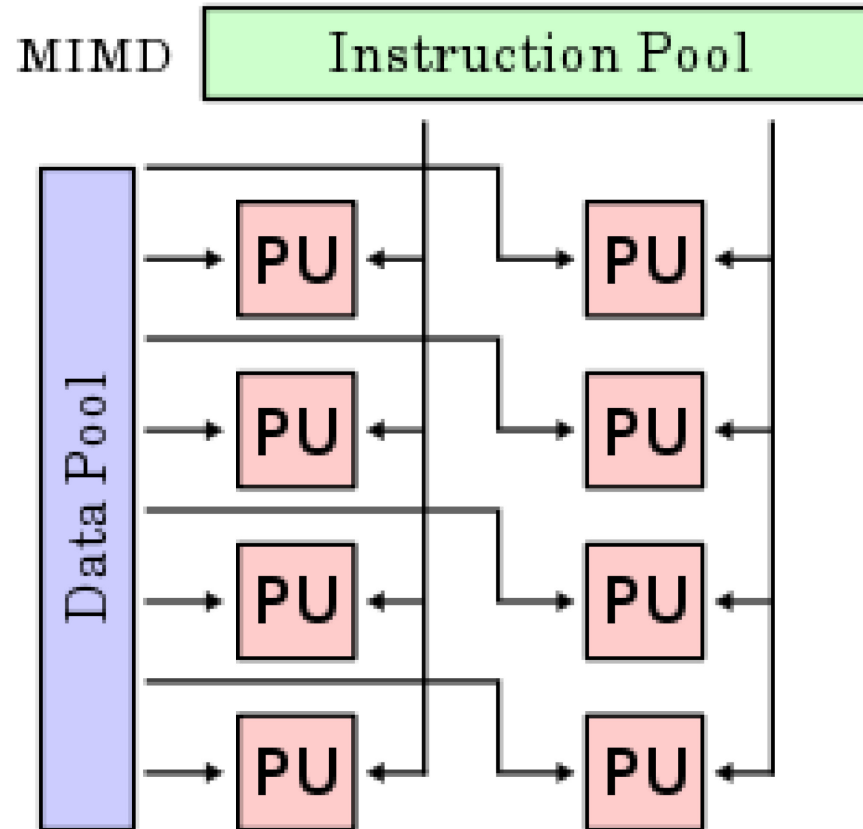
Data pool = Veri havuzu, **Instruction Pool** = Komut Havuzu, **PU** = İşlem birimi (İşlemci, Processing Unit).

MIMD

Multiple Instruction Multiple Data (Çoklu Komut Çoklu Veri)

- İşlem birimlerinin (işlemci) birbirinden bağımsız olarak verileri taşıdığı ve işleme koyduğu paralel (eşzamanlı, koştur) bilgisayar mimarisidir.
- Buna göre, birden çok otonom (kendisi çalışabilen) işlem birimleri farklı komutları farklı veriler üzerinde yürütebilmektedir.
- Sonuçlar elde edilince her bir işlemci bireysel sonucunu merkezi bir yerleşime (belleğe) gönderebilir.
- Buna örnek olarak **dağıtık sistemler** (distributed systems) verilebilir. Bu amaçla, Paylaşımlı bellek uzayı (shared memory space) veya Dağıtılmış bellek uzayı (distributed memory space) kullanılabilir.

MIMD



Anlamları:

Data pool = Veri havuzu, **Instruction Pool** = Komut Havuzu, **PU** = İşlem birimi (İşlemci, Processing Unit).

Küme Komut İşleme (Pipelining)

- Elektrik sinyalleri bir iletken tel içerisinde ışık hızının daha altında bir hızla ilerler. Işık 1 nanosaniye (ns yani saniyenin milyarda biri) yaklaşık 30 cm yol aldığından, CPU'nun, yaklaşık 30 cm uzaklıktaki bir ana bellekten komut çekmesi (Fetch) yaklaşık 2 ns zaman alır.
- Burada, Okuma isteğinin belleğe ulaşması 1 ns, komutun bellekten CPU ulaşması 1 ns. (Komut kod çözme (decoding) için de belirli bir süre geçer). Sonuç olarak bir komutun getirilmesi (Fetch) ve yürütülmesi (Execution) belli bir süre alır.
- Ancak bir bilgisayarın performansını artırmanın tek yolu komutları yürütme hızını artırmak değildir. Gerçek amaç bilgisayarın belli bir sürede gerçekleştirdiği toplam işi, yani **Üretilen iş (Throughput)** değerini artırmaktır.

Küme Komut İşleme (Pipelining)

- Yürütme hızını artırmadan bilgisayarın üretilen işini artırmanın bir yolu da **küme komut işleme**'dir (pipelining, iş hattı veya boruhattı olarak da kullanılır). Bu teknikte bilgisayar makine çevrimindeki adımların örtüşmesine izin verilir. Yani bir komut yürütülürken bir sonraki komutun GETİR (FETCH) adımı başlatılabilir.
- Buna göre, aynı anda birden çok komutun farklı adımları gerçekleştirilebilir. Sonuçta komutların getirilmesi (fetch) ve yürütülmesi (execution) için gereken süre değişmese de bilgisayarda üretilen iş değeri artar.
- Modern bilgisayarlarda birden çok komut eğer birbirlerine bağlı (bağımlı) değillerse aynı anda bu şekilde yürütülebilirler.

Çok işlemcili Mimariler

- **Küme komut işleme** (pipelining) kullanımı, aynı anda birden çok işlemi gerçekleştirerek performans artışı sağlayan paralel işlemenin ilk adımıdır. Ancak, gerçek paralel işleme için birden fazla işlemci birimi gereklidir.
- Bu tür sistemlere **çok işlemcili** veya **çok çekirdekli makine** adı verilir.
- Bugün üretilen bilgisayarların bir çoğu çok işlemcilidir. Bu tür sistemlerde kullanılan yöntemlerden biri, birden fazla işlemci ünitesini aynı ana belleğe bağlamaktır.
- Bu konfigürasyonda (düzenleşim), işlemciler bağımsız olarak çalışabilirken birbirleriyle haberleşmeleri gerektiğinde ortak kullanılan bellek hücrelerine mesaj bırakırlar.
- Örnek olarak büyük bir görevle karşı karşıya kalan bir işlemci, bu görevin bir bölümünü bir program olarak ana belleğe kaydedebilir. Sonrada başka bir işlemciden bu programı yürütmesini isteyebilir.
- Bu durumda, farklı komut dizileri farklı veri kümeleri üzerinde yürütülmüş olur.
- Bu tür mimariler **Çoklu Komut Çoklu Veri (MIMD)** olarak adlandırılır.

Çok işlemcili Mimariler

- Bunun haricinde, tek işlemcili mimariler ise **Tek Komut Tek Veri (SISD)** olarak bilinirler.
- Çok işlemcili makinelerde uygulanan farklı bir mimari ise işlemcilerin aynı komutları kendilerine ait farklı veri kümeleri üzerinde yürütmeleridir. Bu mimariye **Tek Komut Çoklu Veri (SIMD)** denilmekte ve bu tür işlemciler tek bir görevin çok büyük veri kümeleri üzerinde çalıştırılması gerektiği durumlarda daha elverişlidirler.
- **Paralel İşleme** (Parallel Computing) başka bir yaklaşımdır, her biri kendi işlemci ve belleğine sahip olan küçük makineleri birleştirerek büyük bilgisayarlar kurmak amaçlanmaktadır.
- Böyle bir mimaride her bir küçük makine komşu makinelere bağlıdır. Böylelikle tüm sisteme atanan bir görev küçük makinelere bölüştürülebilir. Eğer bilgisayarlardan birine verilen görev birbirinden bağımsız alt görevlere bölünebiliyorsa, o bilgisayar komşularından bu alt görevlerin bazılarını eşzamanlı olarak yapmasını isteyebilir.
- Böylelikle tek bir makinenin yapabileceğinden çok daha kısa bir sürede görev tamamlanmış olur.

Çok çekirdekli işlemciler

- Teknoloji tek bir silikon çipin içerisine çok daha fazla devrenin yerleştirilmesine imkan tanıdıkça, bilgisayar bileşenleri arasındaki fiziksel mesafede oldukça azaldı. Bu nedenle, tek bir çipin içerisine hem CPU hem de ana bellek yerleştirilebildi. Buna “**çip içi sistemler**” (**SoC-System on Chip**) bir örnek olarak verilebilir.
- Bunun haricinde, tek bir cihaz içerisine aynı devrenin bir çok kopyası yerleştirilebilir. Bu yaklaşımla, başlangıçta birden çok birbirinden bağımsız lojik (mantık) kapısı veya Flip-Flop devresi aynı çipin içerisinde olacak şekilde üretilmiştir.
- Bugünkü teknolojimizde, çok çekirdekli işlemcilerde, aynı çipin içerisinde birden fazla işlemcinin ve paylaşılmış bir önbelleğin (cache) bulunduğu CPU’lar üretebiliyoruz. İki işlemci birimi içeren çok çekirdekli işlemciler çift-çekirdekli (dual-core), dört işlemci birimi içerenler ise dört çekirdekli (Quad-core) olarak biliniyorlar. Şu an altı çekirdekli olanlar mevcuttur.
- Bu tarz MIMD mimarili sistemler rahatlıkla kurulabilmektedir.

Kaynaklar

[Kaynak 1] Tanenbaum, (2008) **Modern Operating Systems** 3 ed, 2008, Prentice-Hall, Inc. Book.

[Kaynak 2] J. Glenn Brookshear, Dennis Brylow, (2016), **Bilgisayar Bilimine Giriş**, 12 Ed., (Türkçe çevirisi), Nobel Yayınevi, ISBN : 6053203612.