

MAKİNE DİLİ VE BROOKSHEAR MİMARİSİ



BURSA TEKNİK
ÜNİVERSİTESİ

içİNDEKİLER

Kısaca ana hatlar

- 1.Bilgisayar Mimarisi Temelleri (CPU, ALU, CU, Register)
- 2.Makine Dili Kavramı ve Instruction Set
- 3.Brookshear Makinesi ve Özellikleri
- 4.Komut Yapısı(Op-Code ve Operand)
- 5.Programın Yürütülmesi(Fetch-Decode-Execute)
- 6.Uygulama
- 7.Sonuç ve Değerlendirme

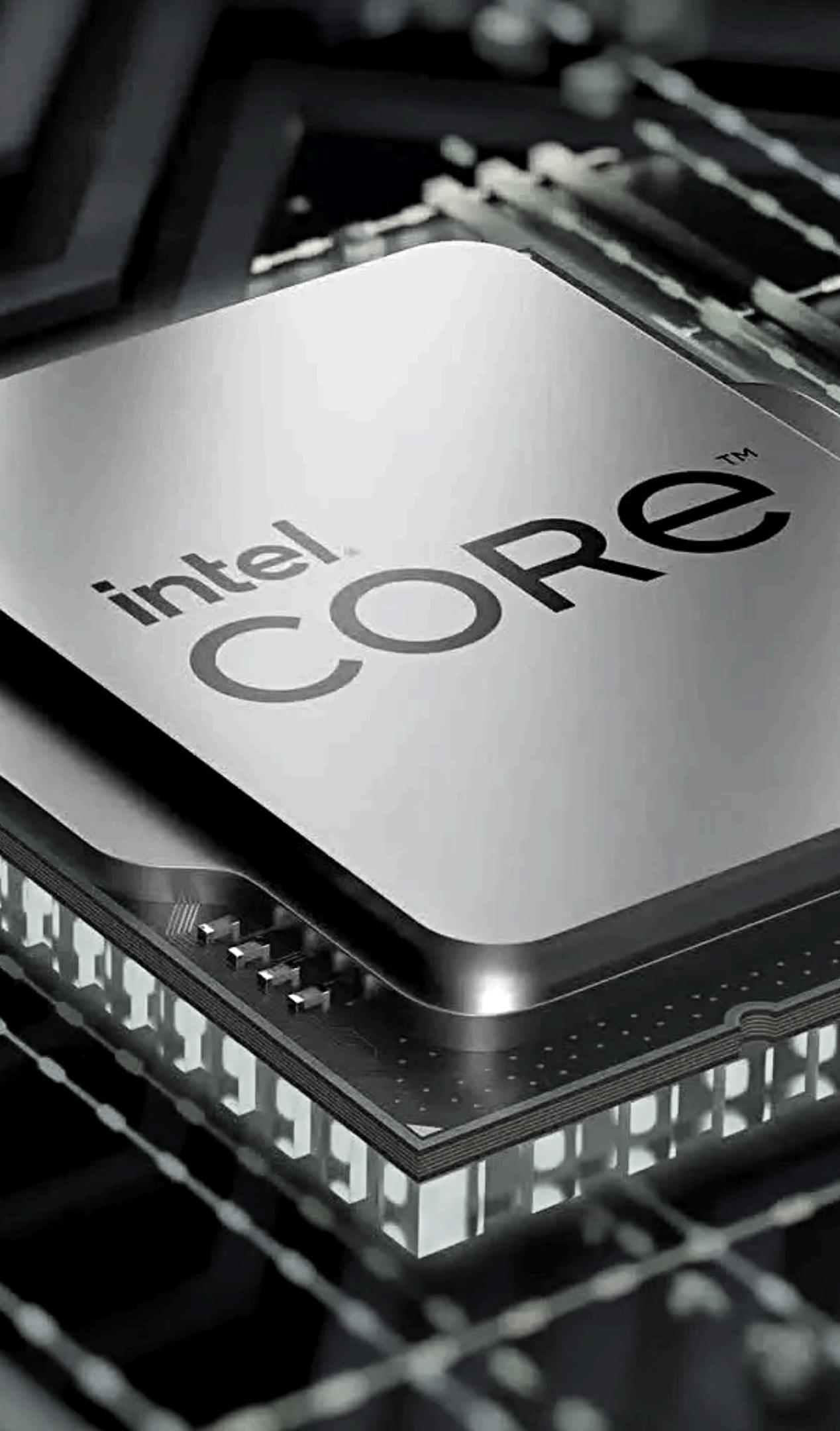
Bilgisayar Mimarisi

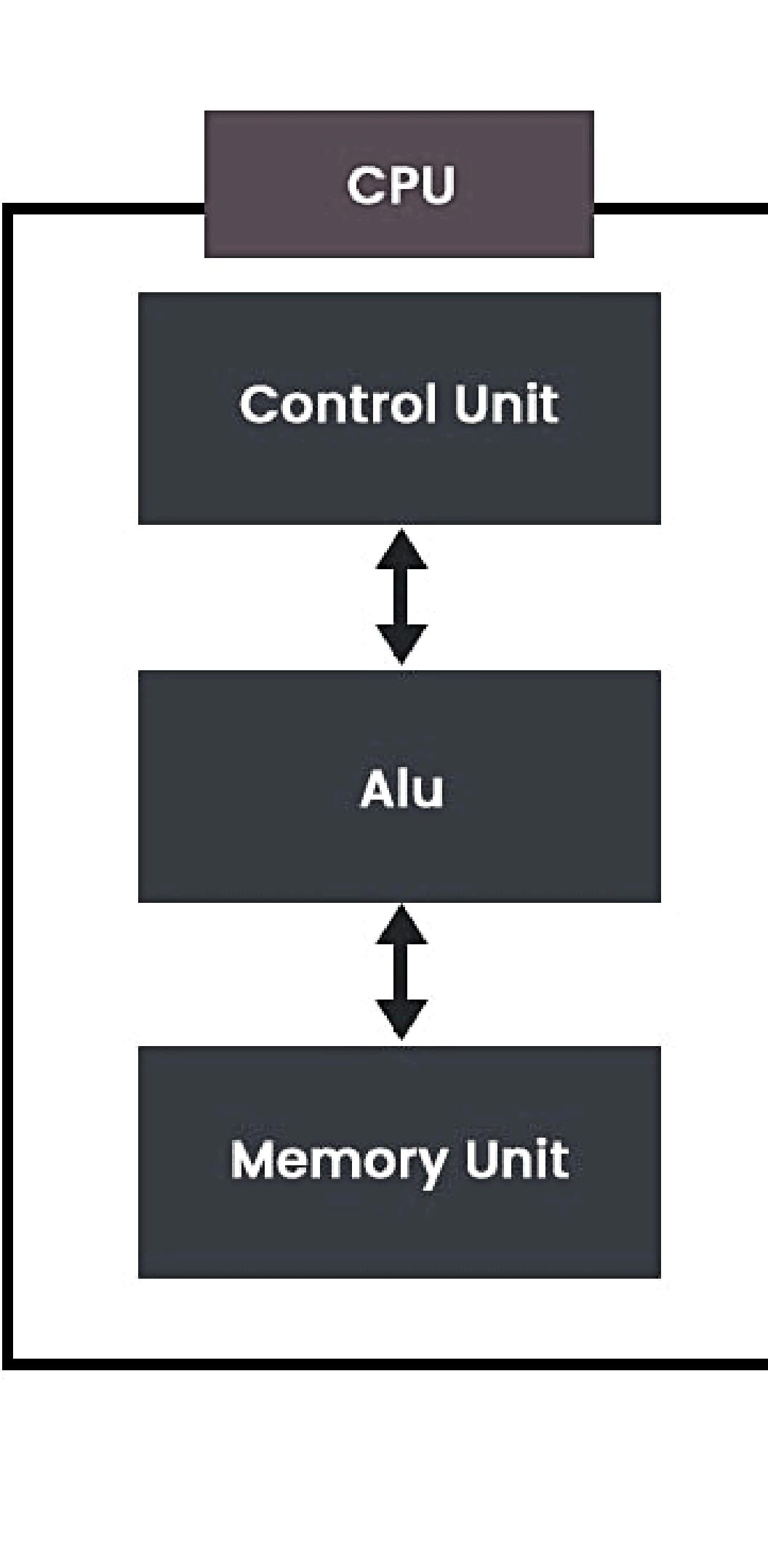
Merkez İşlem Birimi(CPU)

CPU, bilgisayarın donanım ve yazılım birimleri arasındaki veri akışını kontrol eden ve komutları işleyen en temel donanım bileşenidir. Genellikle “bilgisayarın beyni” olarak adlandırılır.

Temel görevleri:

- **komut işleme:** Yazılımlardan gelen karmaşık komutları makine diline çevirerek uygular.
- **veri yönetimi:** Bellekten gelen verileri alır, işler ve sonuçları tekrar belge veya çıkış birimlerine gönderir.
- **koordinasyon:** Bilgisayarın diğer tüm parçalarının (RAM, ekran kartı, depolama vb.) bir uyum içinde çalışmasını sağlar.





CPU iç Mimarisinin Temel Bileşenleri

- **ALU (Aritmetik Mantık Birimi)**

Matematiksel ve mantıksal işlemlerin mutfağıdır.

- **Control Unit (Kontrol Birimi)**

Komutları çözen ve birimleri yöneten beyin
kısımidır

- **Registers (Yazmaçlar)**

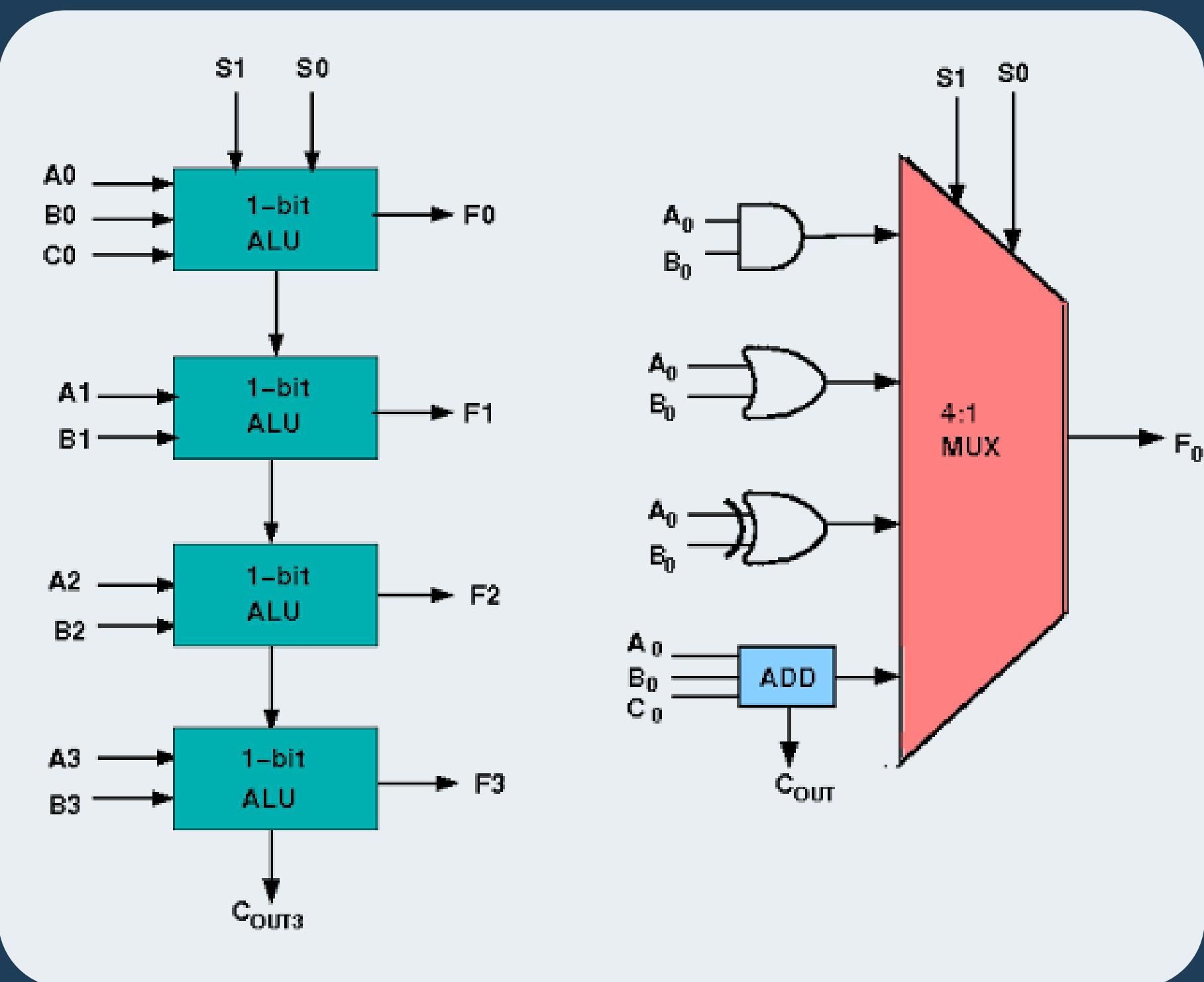
Verilerin işlem anında tutulduğu çok hızlı bellek
alalnlarıdır



ALU (Aritmetik Mantık Birimi)

Aritmetik Mantık Birimi (ALU) Nedir?

- **Aritmetik İşlemler:** CPU'nun ihtiyaç duyduğu toplama, çıkarma, çarpma ve bölme gibi temel matematiksel işlemlerin yapıldığı bölümdür.
- **Mantıksal İşlemler:** VE (AND), VEYA (OR), DEĞİL (NOT) gibi mantıksal kıyaslamaları gerçekleştirir.
- **Karar Verme:** İki veriyi karşılaştırarak ($A > B$ mi? $A == B$ mi?) programın akışına karar verir
- **Veri Yolu:** İşlem görecek veriler yazmaçlarından (registers) ALU'ya gelir, sonuç yine bir kaydediciye yazılır.



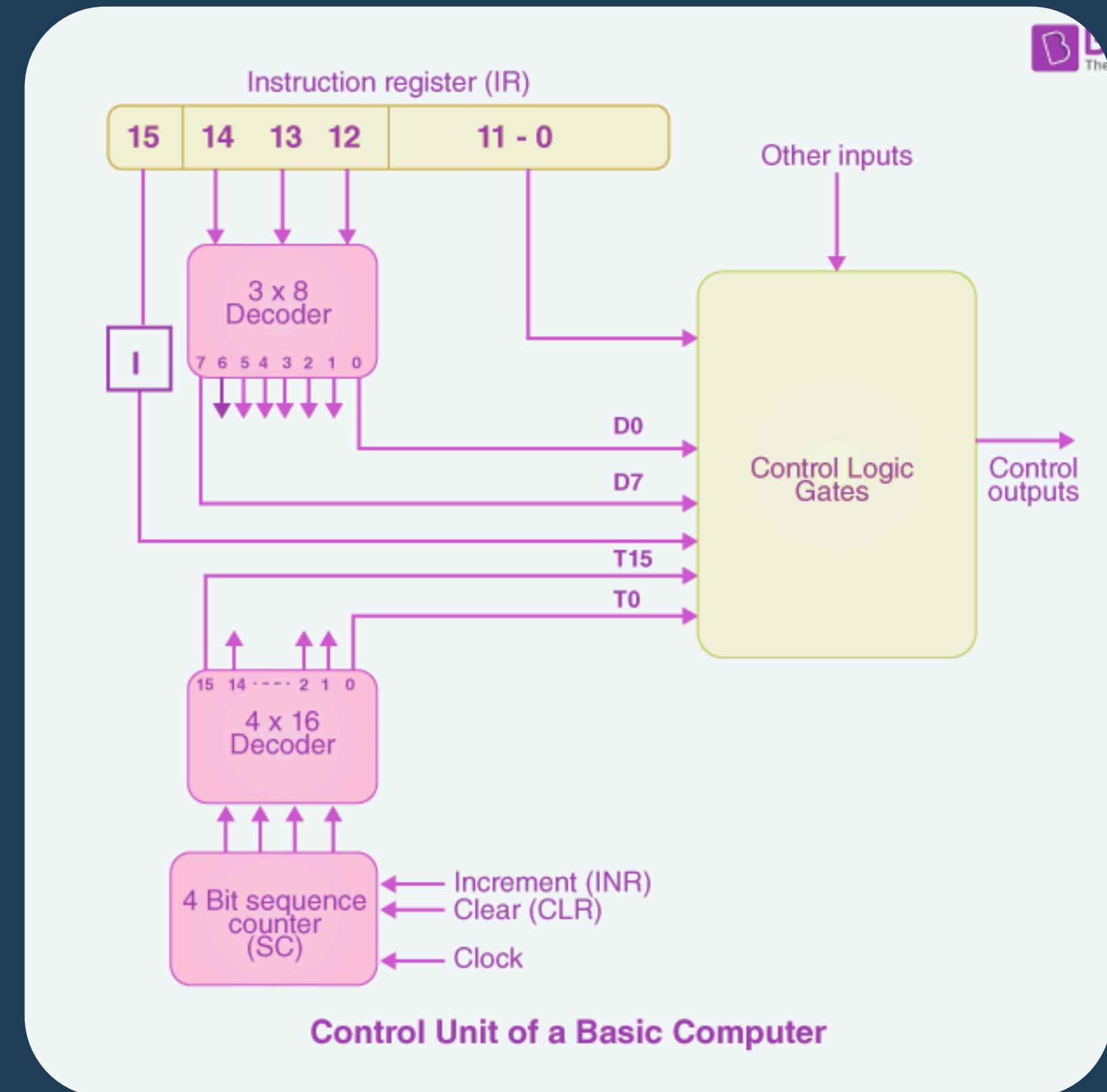
Control Birimi (Control Unit - CU)

Control Birimi (CU) Nedir?

Control Birimi, işlemci içindeki tüm faaliyetleri koordine eden ve komutların doğru sırayla yürütülmesini sağlayan birimdir.

Temel Görevleri:

- **Komut Getirme (Fetch):** Bir sonraki çalıştırılacak komutu ana bellekten (RAM) alır.
- **Komut Çözme (Decode):** Gelen makine dili komutunun (örneğin "2103") ne anlama geldiğini ve hangi işlemin yapılacağını analiz eder.
- **Sinyal Gönderimi:** ALU'ya ve yazmaçlara işlemin başlaması için gerekli elektriksel kontrol sinyallerini gönderir.
- **Zamanlama:** Tüm işlemlerin saat vuruşlarıyla (Clock Speed) uyumlu bir şekilde gerçekleşmesini sağlar.



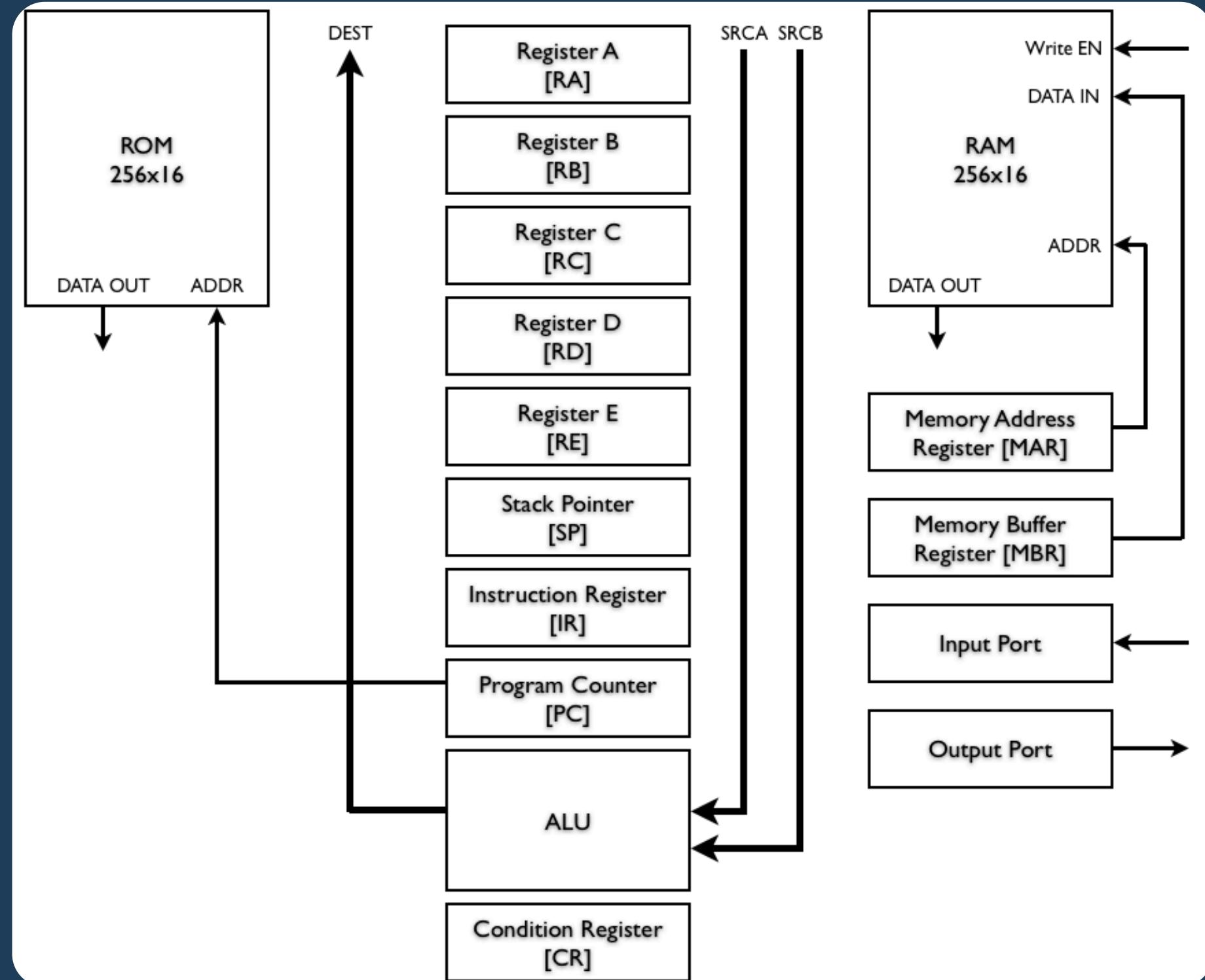
Yazmaçlar (Registers)

Yazmaçlar (registers) Nedir?

Yazmaçlar, işlemcinin içerisinde bulunan, verilerin ve komutların işlem gördüğü sırada çok kısa süreliğine depolandığı, erişim hızı en yüksek olan bellek birimleridir.

Önemli Yazmaç Türleri

- **Genel Amaçlı Yazmaçlar (General Purpose Registers):** İşlem sırasında geçici verileri (sayıları, sonuçları) tutar. (Brookshear makinesinde 16 tanedir).
- **Program Sayacı (Program Counter - PC):** Bir sonraki çalıştırılacak komutun bellekteki adresini tutan özel kaydedicidir.
- **Komut Yazmacı (Instruction Register - IR):** O anda işlenmeyece olan güncel komutu tutar.



Ana Bellek (RAM) ve Hexadecimal Adresleme

Bellek Hücreleri ve Hexadecimal Düzen

Brookshear mimarisinde ana bellek, verilerin ve komutların işlemci tarafından işlenmeden önce beklediği “posta kutuları” dizisi gibidir.

Hücre Yapısı: Bellek, her biri 8 bit (1 Byte) veri saklayabilen bağımsız hücrelerden oluşur.

Kapasite ve Adresleme: Toplamda 256 hücre bulunur. Bu hücreler, karmaşıklığı önlemek için Hexadecimal (16'lık) sayı sistemi ile 00'dan FF'ye kadar adreslenir.

Erişim: İşlemci, bir veriye ulaşmak istediğiinde sadece o verinin bulunduğu hücrenin adresini (Örn: A5 veya 3C) kullanır.

Esneklik: Aynı bellek alanı hem program komutlarını hem de üzerinde işlem yapılacak sayısal değerleri saklayabilir.

Adres	Veriler
0x0	1 Bayt
...	1 Bayt
0x1F40	1 Bayt
0x1F41	1 Bayt
0x1F42	1 Bayt
0x1F43	1 Bayt
0x1F44	1 Bayt
0x1F45	1 Bayt
0x1F46	1 Bayt
0x1F47	1 Bayt
0x1F48	1 Bayt
...	1 Bayt

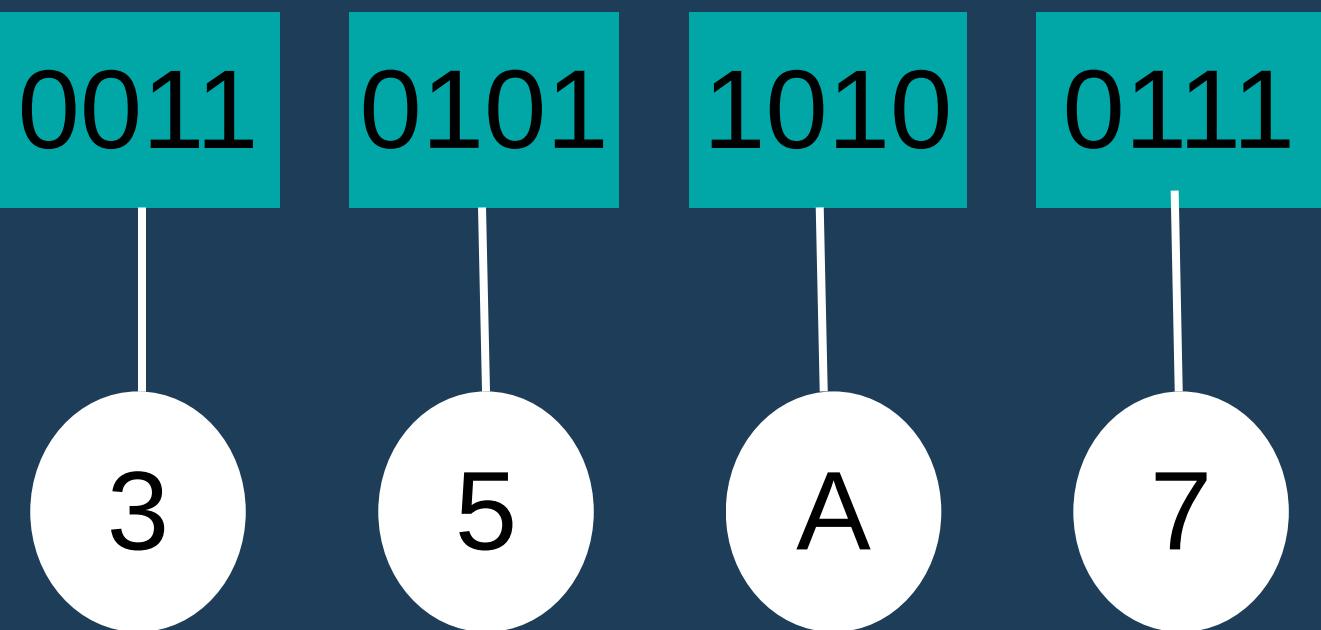
Makine Dili ve Komut (Instruction) Kavramı



Komut Yapısı (Instruction Format)

Makine dili, bir işlemcinin doğrudan anlayabildiği ve yürütebildiği en düşük seviyeli programlama dilidir. Brookshear mimarisinde her bir işlem, belirli bir formatta yazılmış "komutlar" ile ifade edilir.

- **Komut Nedir?:** İşlemciye ne yapması gerektiğini söyleyen bit dizileridir. Brookshear makinesinde her komut 16 bit (2 Byte) uzunluğundadır.
- **Hexadecimal Gösterim:** 16 bitlik karmaşık ikili sayıları (0 ve 1) okumak zor olduğu için, komutlar genellikle 4 haneli Hexadecimal kodlar şeklinde yazılır (Örn: 156A).
- **Instruction Cycle (Komut Döngüsü):** Bir komutun çalışması için daha önce gördüğümüz Fetch (Getir), Decode (Çöz) ve Execute (Yürüt) aşamalarından geçmesi gereklidir.



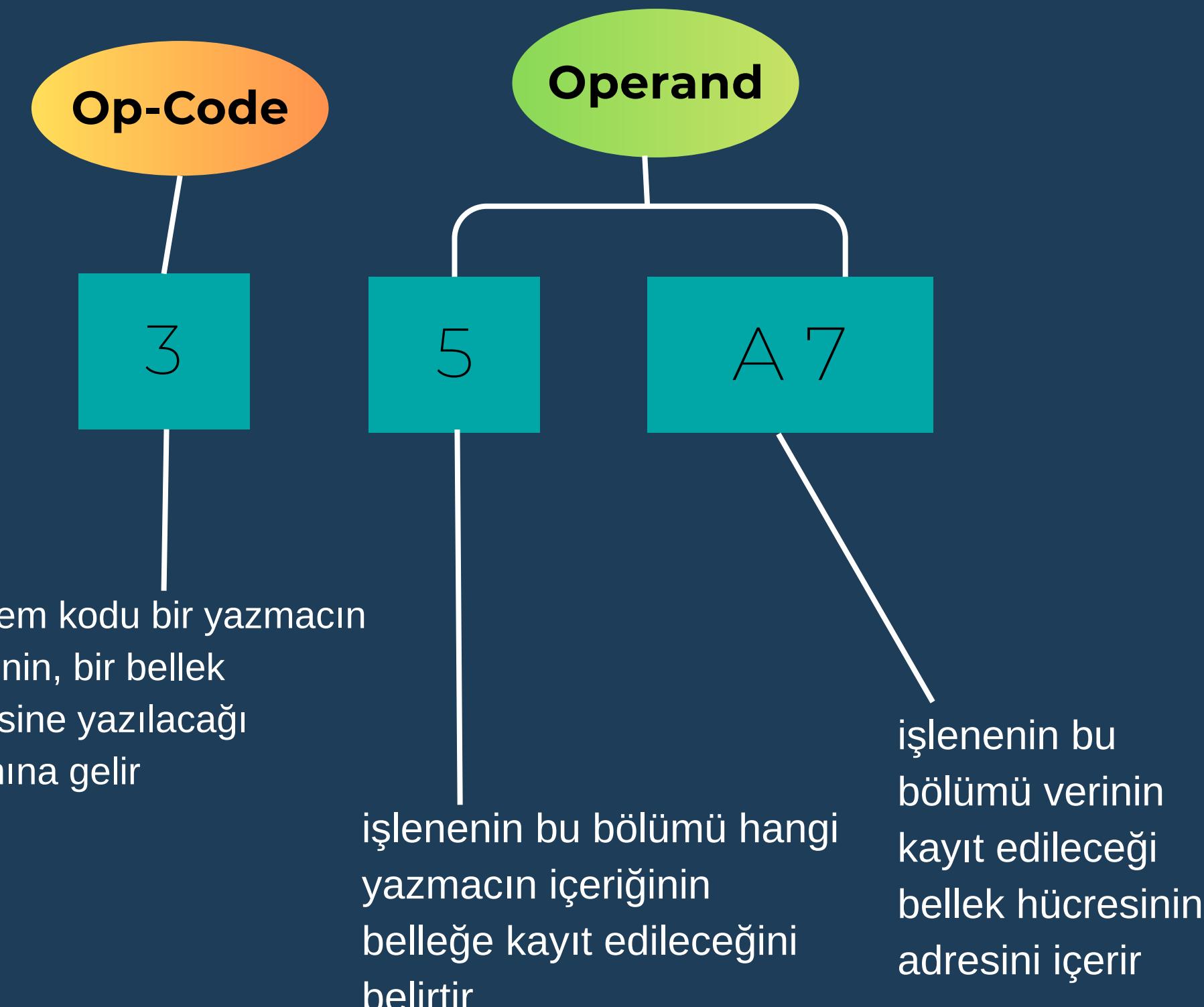
Op-Code (İşlem Kodu) ve Operand (İşlenen)

Op-Code ve Operand Nedir?

OP-CODE (İşlem Kodu): Komutun ilk karakteridir. İşlemciye “topla”, “yükle” veya “dallan” gibi ne yapması gerektiğini söyleyen ana emirdir.

OPERAND (İşlenen): Komutun kalan 3 karakteridir. Bu kısım, işlemin kiminle yapılacağını söyler:

- Hangi kaydedicinin (register) kullanılacağı
- Hangi bellek adresine (RAM) gidileceği
- Hangi sabit değerin işleneceği





Brookshear Makinesi ve Mimarisi

BROOKSHEAR MAKİNESİ NEDİR?

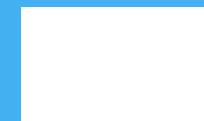
Dr.J.Glenn Brookshear tarafından bilgisayar mimarisini öğretmek amacıyla geliştirilmiş sanal bir yapıdır. Von Neumann mimarisini temel alır, veri ve komutlar aynı bellektedir.

256 hücreden (00'dan FF'ye kadar adreslenmiş) oluşur. 16 adet genel amaçlı (0-F) arası kaydedici (register) barındırır. Karmaşıklıktan uzak, en temel 12 işlem üzerine kuruludur.

CPU

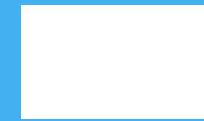
registers

0



Program Counter

1



2



•
•
•

Instruction Register

F



RAM (ANA BELLEK)

Address

A0

15

A1

6C

A2

16

A3

6D

A4

50

A5

56

A6

30

A7

6E

A8

C0

A9

00

12 TEMEL MAKİNE DİLİ KOMUTU

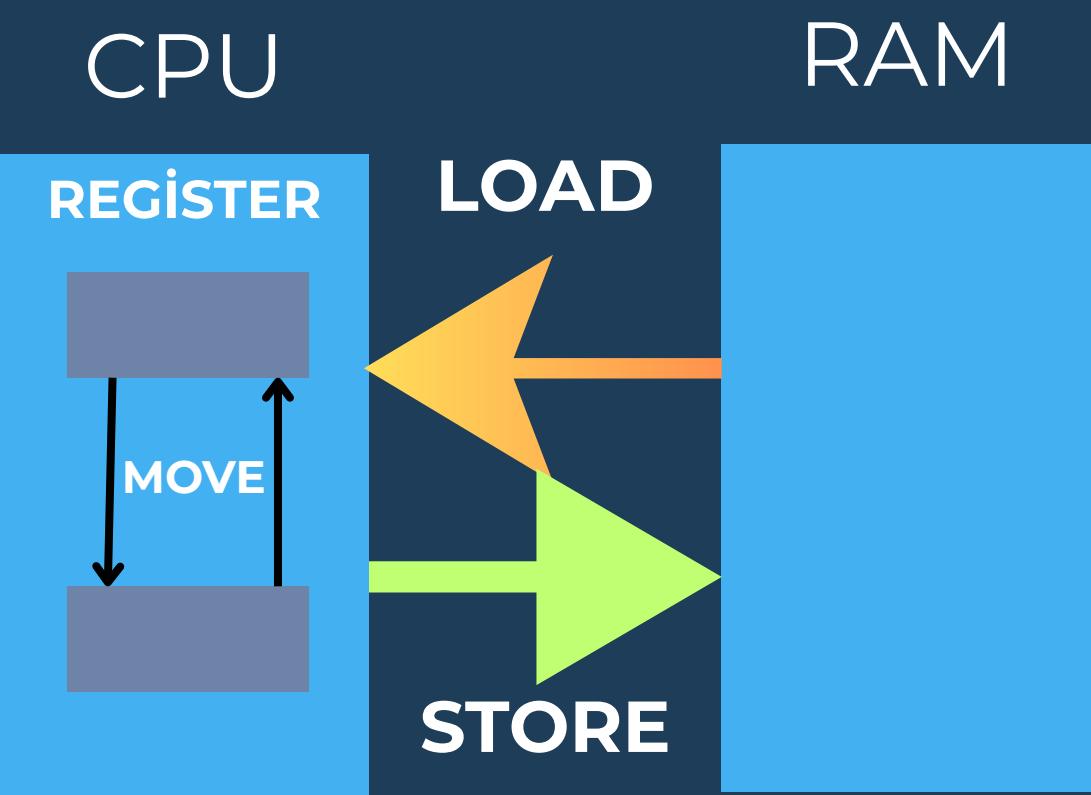
OP-CODE	İŞLEM	AÇIKLAMA
1	LOAD (YÜKLE)	BELLEK HÜCRESİNDEKİ VERİYİ BİR YAZMACA (REGISTER) KOPYALAR
2	LOAD	YAZMACA DOĞRUDAN BELİRLİ BİR DEĞERİ YÜKLER
3	STORE (SAKLA)	YAZMAÇTAKİ VERİYİ BİR BELEK HÜCRESİNÉ KOPYALAR (SAKLAR)
4	MOVE (TAŞI)	BİR YAZMAÇTAN DİĞER BİR YAZMACA VERİ AKTARIR
5	ADD (TOPLA)	İKİ YAZMACI "İKİNİN TÜMLEYENİ" YÖNTEMİYLE TOPLAR
6	ADD	İKİ YAZMACI "KAYAN NOKTALI" OLARAK TOPLAR

OP-CODE	İŞLEM	AÇIKLAMA
7	OR (VEYA)	İKİ YAZMACA MANTIKSAL “VEYA” İŞLEMİ UYGULAR
8	AND (VE)	İKİ YAZMACA MANTIKSAL “VE” İŞLEMİ UYGULAR
9	XOR	İKİ YAZMACA XOR İŞLEMİ UYGULAR
10	ROTATE (DÖNDÜR)	BİR YAZMAÇTAKİ BİTLERİ SAĞA DOĞRU DAİRESEL DÖNDÜRÜR
11	JUMP (ATLA)	EĞER YAZMAÇ İÇERİĞİ BELİRTİLENLE AYNIYSA, VERİLEN ADRESE ATLAR
12	HALT (DURDUR)	PROGRAMIN ÇALIŞMASINI TAMAMEN SONLANDIRIR

Veri Transferi Komutları

Bellek ve İşlemci Arasındaki Trafik

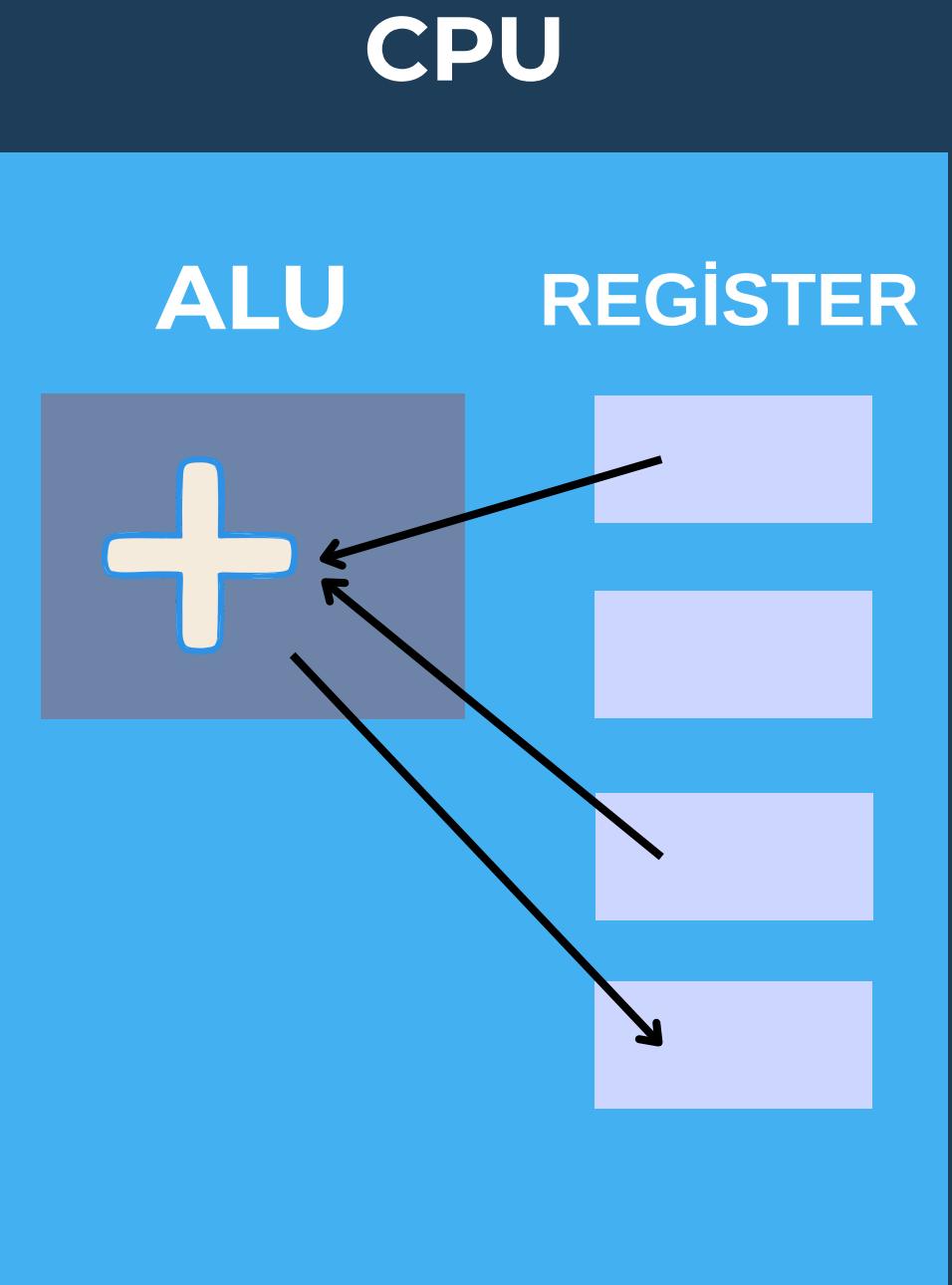
- **komut 1 (LOAD):** Belirtilen bellek hücresindeki veriyi bir yazmaca (register) kopyalar. Örnek: 14A3 kodu için A3 adresindeki veriyi 4 numaralı yazmaca (register) yükle
- **komut 2 (LOAD):** Bir yazmaca doğrudan bir değer yüklemek için kullanılır. Örnek: 2310 kodu için 3 numaralı yazmaca “10” değerini yaz.
- **komut 3 (STORE):** Bir yazmacıtaki veriyi ana bellekteki bir hücreye kopyalar. Örnek: 35B2 kodu için 5 numaralı yazmacıtaki veriyi B2 Adresli hücreye yaz.
- **komut 4 (MOVE):** Bir yazmacıtaki veriyi başka bir yazmaca aktarır. (kopyalama yapar kaynak silinmez) Örnek: 40A1 kodu için A (16'lık sistemde 10) numaralı yazmacıtaki veriyi 1 numaralı yazmaca kopyalar.



Aritmetik İşlem Komutları

Toplama İşlemi ve Veri Tipleri

- **komut 5 (ADD-tamsayı):** İki yazmaçtaki veriyi “İkinin Tümleyeni” yöntemine göre toplar ve sonucu hedef yazmaca yazdırır. Örnek: 5321 kodu için 2 ve 1 numaralı yazmaçlardaki veriyi ikinin tümleyeni yöntemiyle topla ve sonucu 3 numaralı yazmaca yazdır.
- **komut 6 (ADD-ondalıklı sayı):** Bu komut ondalıklı sayıları toplamak için kullanılır. Örnek: 6321 kodu için 1 ve 2 numaralı yazmaçlardaki verleri kayan ondalıklı sayı gibi topla ve 3 numaralı yazmaca yazdır.
- Bu işlemler ALU'da (Aritmetik Mantık Birimi) gerçekleştirilir.



İkinin Tümleyeni Yöntemi

İkiye Tümleyen	10'lu gösterimi
0111	7
0110	6
0101	5
0100	4
0011	3
0010	2
0001	1
0000	0
1111	-1
1110	-2
1101	-3
1100	-4
1011	-5
1010	-6
1001	-7
1000	-8

2's Complement

Addition

$$\begin{array}{r} +14 \rightarrow 01110 \\ -5 \rightarrow +11011 \\ +9 \rightarrow \hline \end{array}$$

Subtraction

$$\begin{array}{r} +9 \rightarrow 01001 \\ -6 \rightarrow +11010 \\ +3 \rightarrow \hline \end{array}$$

Kayan Nokta Gösterimi

3 bit kullanılan fazlalık gösterim sistemi

111	3
110	2
101	1
100	0
001	-3
010	-2
011	-1
000	-4



işaret biti

0 = +
1 = -

fazlalık
gösterim
sisteme
göre
110 = 2
bu sayıyı 2
kaydıracağız

asıl
değerimiz
bu sayıyı 2
kaydıracağız

$$10.11 = 2^{3/4}$$

$$1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2}$$

Mantıksal İşlem Komutları

Kapı Mantığı ve Karar Verme

- **komut 7 (OR-VEYA):** İki yazmaçtaki bitleri karşılaştırır. Aynı konumdaki bitlerden herhangi biri 1 ise sonuç 1 olur.
- **komut 8 (AND-VE):** İki yazmaçtaki bitleri karşılaştırır. Aynı konumdaki bitlerden biri 0 ise sonuç 0 olur.
- AND işlemi ile “Maskeleme” yapılır. Örneğin 10110110 sayısının sadece ilk 4 bitini sıfırlamak istiyoruz, bunun için korumak istediğimiz yere 1, silmek istediğimiz yere 0 yazarız . Burdan sayıımız 00001111 haline gelir. Daha sonra elde edilen bu sayıyla başlangıçtaki sayıımızı AND işlemine sokarız sonuç 00000110 olur yani ilk 4 bit sıfırlanmış olur.
- **komut 8 (XOR):** Bitler birbirinden farklısa sonucu 1 yapar, aynıysa 0 yapar.
- Veri şifreleme veya bir yazmacın içeriğini tamamen sıfırlamak için kullanılır.

a	b	out
0	0	0
0	1	0
1	0	0
1	1	1

a	b	out
0	0	0
0	1	1
1	0	1
1	1	1

a	b	out
0	0	0
0	1	1
1	0	1
1	1	0

Bitleri Döndürme Komutu (ROTATE)

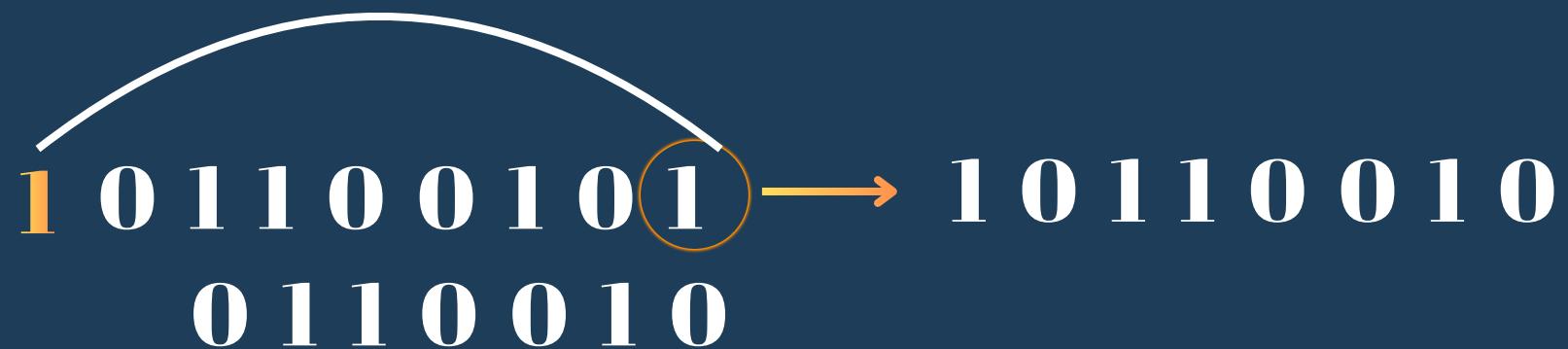
Sayıyı Sağa Doğru Kaydırma

Bir yazmacın içindeki bit dizilimini belirtilen sayı kadar sağa doğru dairesel olarak kaydırır. En sağdaki (en düşük değerli) bit kaybolmaz, dönüp en sol başa (en yüksek değerli konuma) geçer.

Örnek: A502 komutu için 5 numaralı yazmaca ait bitleri 2 kez sağa döndürür.

Veri şifreleme, seri iletişim ve hızlı çarpma/bölme işlemlerinde kullanılır.

0x65 Hexadecimal sayının bir bit sağa döndürülmesi

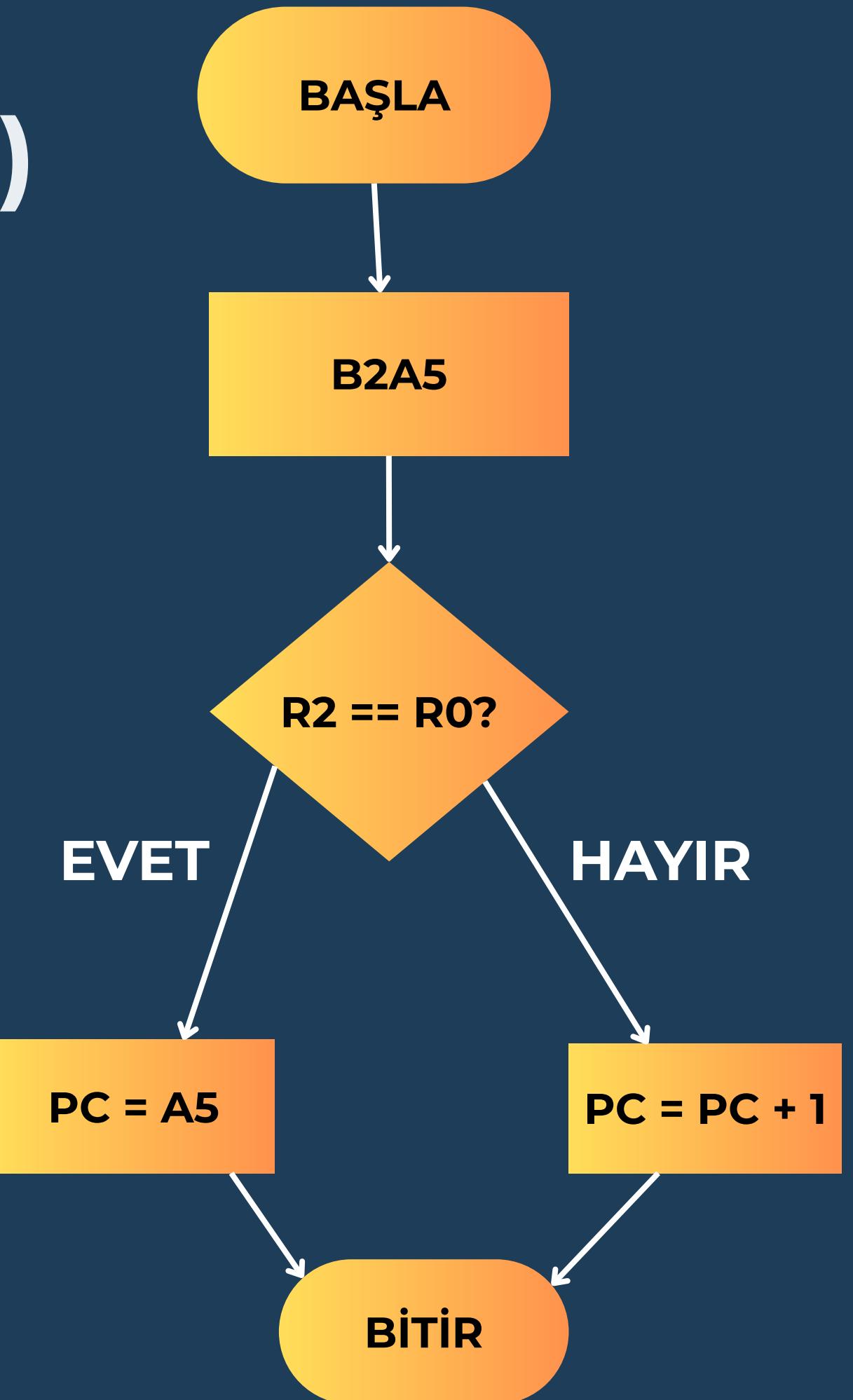


bu işlemi 8 kere tekrar edersek aynı bit desenini elde ederiz

Koşullu Atlama Komutu (JUMP)

Karar Verme ve Dallanma Mekanizması

- Programın normal, ardışık giden akışını değiştirerek başka bir bellek adresine “atlanmasını” sağlar. Eğer belirtilen yazmacın içeriği 0 numaralı yazmaçtaki değerle aynıysa atlama gerçekleşir. Eğer değerler aynı değilse işlemci atlama yapmaz ve bir sonraki saturdaki komutla yoluna devam eder.
- Örnek: B2A5 komutu için eğer 2 numaralı yazmaçtaki veri 0 numaralı yazmaçtaki veri ile aynıysa program sayacını (PC) A5 adresine yönlendirir.



İşlemi Sonlandırma Komutu (HALT)

Programın Başarıyla Tamamlanması

- Brookshear makinesinde programın yürütülmesini durdurulan son komuttur. Kontrol birimine artık "Getir" döngüsünü durdurması gerektiğini söyler.
- Komut **C 0 0 0** şeklindedir.



Peki Tüm Bu Komutlar İşlemci İçinde Nasıl Bir Döngüyle Çalışır?

Program Yürütme



Özel amaçlı iki register tarafından kontrol edilir.

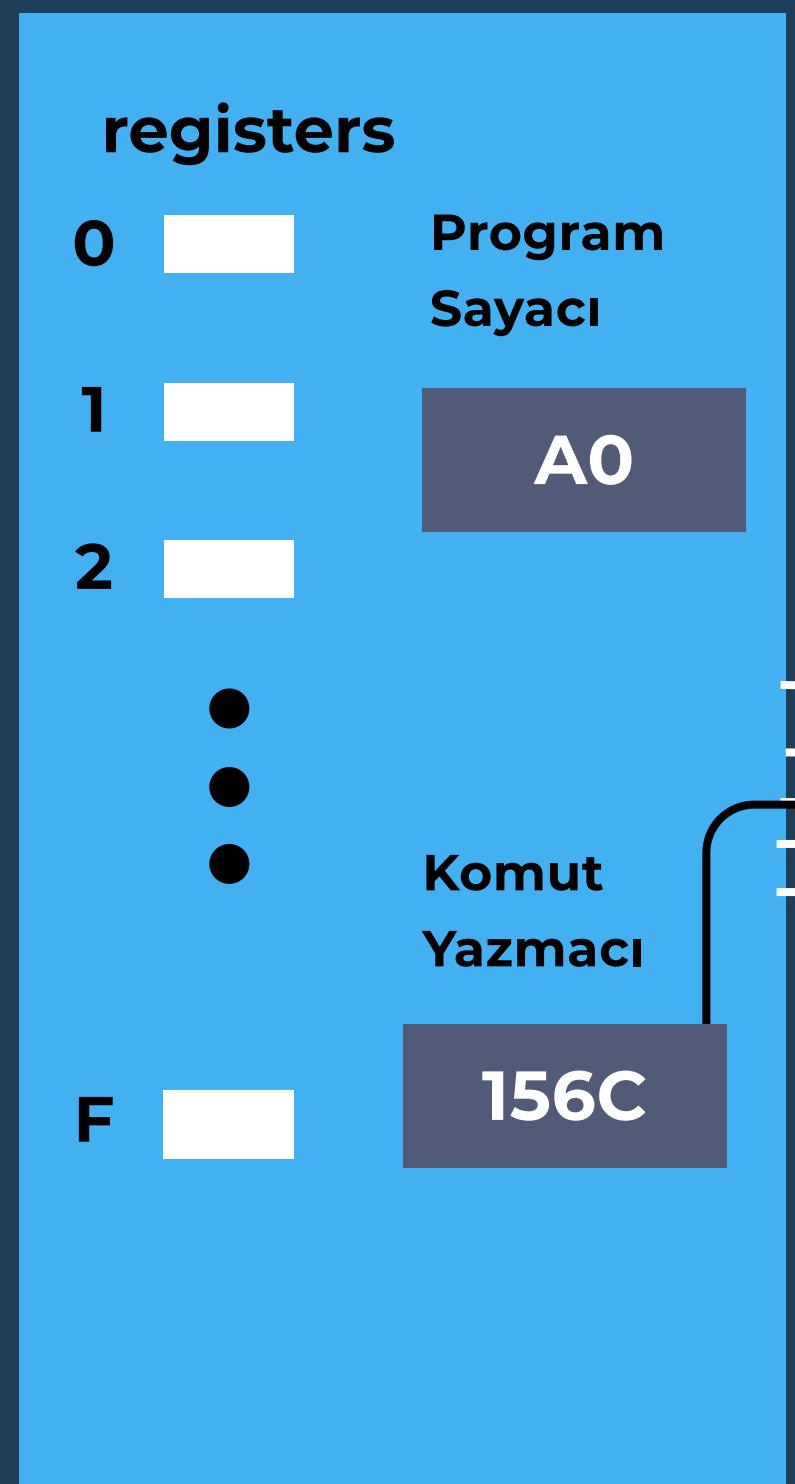
- **Instruction Register (Komut Yazmacı):** Mevcut komutu tutar.
- **Program Counter (Program Sayacı):** Bir sonraki komutun adresini tutar.

Makine Döngüsü

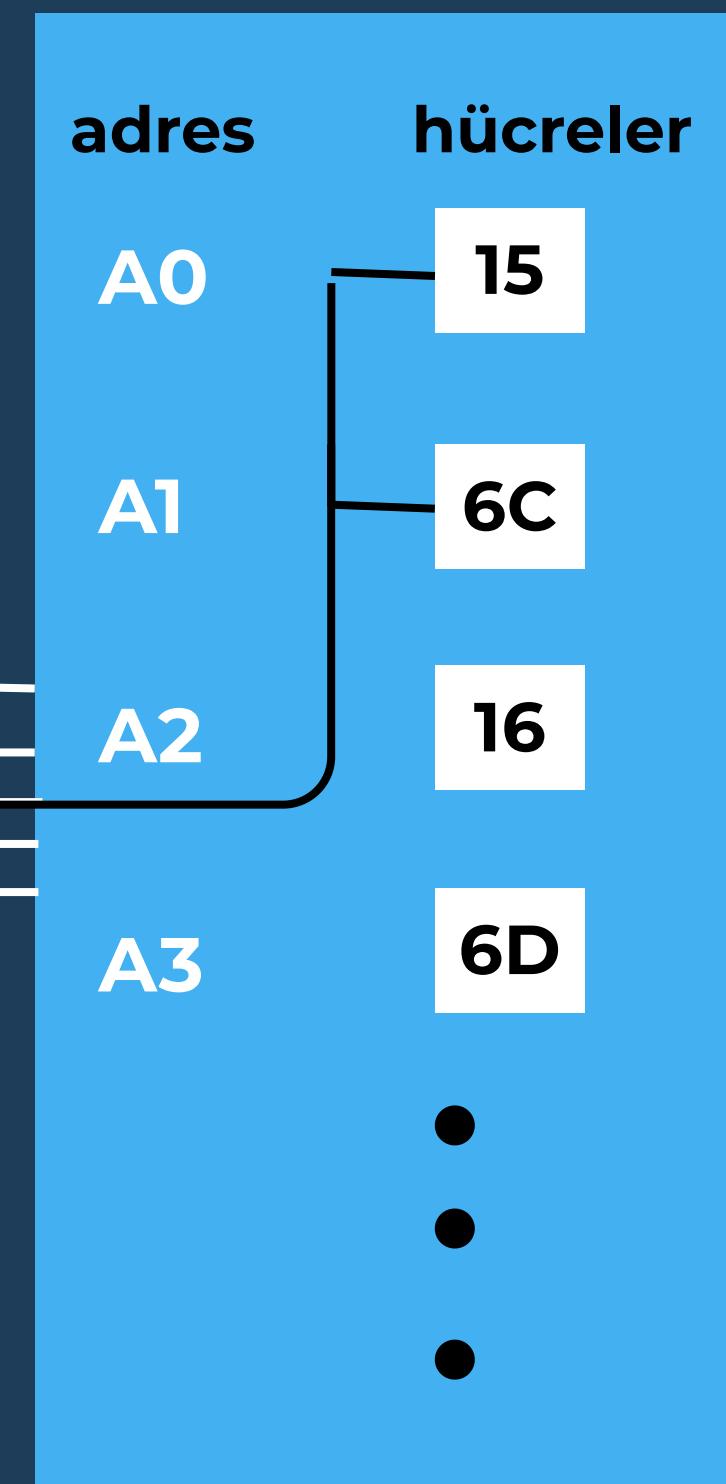
Üç adımı takip eder.

- **Getir (Fetch):** Program sayacındaki (PC) adrese gidilir, komut bellekten alınır ve Komut Yazmacına kopyalanır. Ardından PC bir sonraki adres için arttırılır.
- **Çöz (Decode):** Komut yazmacına gelen 16 bitlik veri kontrol birimi (CU) tarafından analiz edilir. Op-code ve operandlar ayırtılı olarak hangi birimin (ALU, yazmaçlar vb.) çalışacağı belirlenir.
- **Yürüt (Execute):** Belirlenen işlem (toplama, veri taşıma, atlama) gerçekleştirilir. İşlem biter bitmez döngü en başa döner.

CPU



RAM



CPU



RAM



Özet: Bilgisayarın Çalışma Mantığı

Brookshear Makinesi Bize Neler Öğretti?

- **Evrensel Döngü:** Modern işlemciler ne kadar hızlı olursa olsun, hala temel Fetch-Decode-Execute (Getir-Çöz-Yürüt) döngüsüyle çalışır.
- **Donanım ve Yazılım Köprüsü:** Bir yazılımın donanım üzerinde nasıl fiziksel birer komuta (0 ve 1) dönüştüğünü ve adreslendiğini gördük.
- **Hassas Hesaplama:** Bilgisayarın tamsayıları İkinin Tümleyeni, ondalıklı sayıları ise Kayan Nokta yöntemiyle işleyerek hata payını nasıl yönettiğini kavradık.
- **Kontrolün Gücü:** JUMP ve HALT gibi komutlarının, bir makineyi sadece hesap makinesi olmaktan çıkarıp "karar veren" bir bilgisayara dönüştürdüğünü anladık.



**BU ÇALIŞMA, BROOKSHEAR MAKİNESİ VE TEMEL BİLGİSAYAR MİMARİSİ
PRENSİPLERİ ÜZERİNE KAPSAMLI BİR İNCELEME SONUCU
HAZIRLANMIŞTIR.**

Hazırlayan: Ahşen AKYOL

Öğrenci Numarası: 25360859073

Ders: Bilgisayar Mühendisliğine Giriş