BURSA TEKNİK ÜNİVERSİTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

MAKİNE DİLİ VE BROOKSHEAR MİMARİSİ

BLM101 – Bilgisayar Mühendisliğine Giriş

SILA İPEK
24360859018

İçindekiler

Bilgisayar Mimarisi

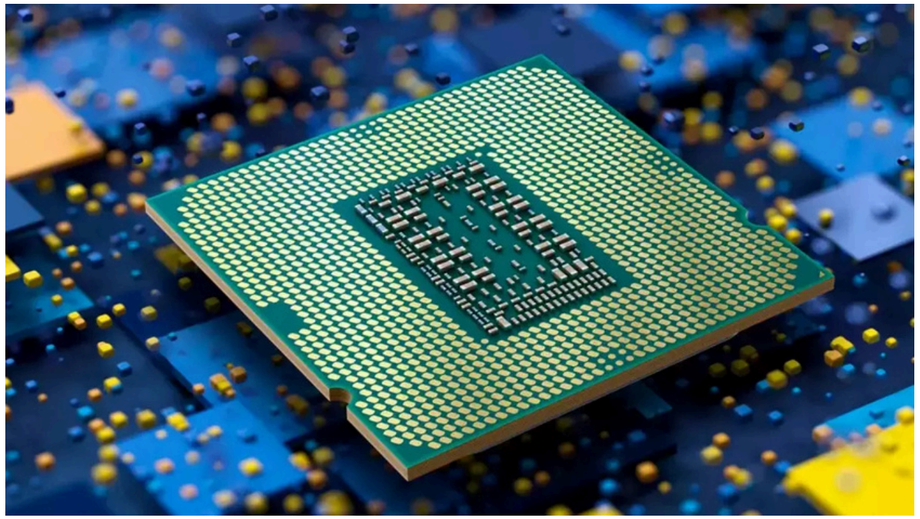
- Bilgisayarda verilerin işlenmesini kontrol eden birime Merkezi İşlem Birimi (CPU) veya işlemci denir.
- Geçmişte CPU'lar çok büyük yapılara sahipken, günümüzde teknolojinin gelişmesiyle oldukça küçülmüştür.
- Masaüstü ve dizüstü bilgisayarlarda işlemciler anakarta takılan yaklaşık 5×5 cm boyutlarında paketlerdir.
- Mobil cihazlarda ise bu daha küçük işlemcilere mikroişlemci adı verilir.

Merkezi İşlem Birimi (CPU)

- CPU, bilgisayar içindeki verilerin manipülasyonunu ve kontrolünü sağlayan devre sistemidir..Bilgisayarda çalışan tüm programların komutlarını alır, yorumlar ve çalıştırır.

CPU'nun temel görevleri:

- Program komutlarını bellekten almak,
- Komutları çözümlemek ,
- Aritmetik ve mantıksal işlemleri gerçekleştirmek,
- Diğer donanım birimlerini kontrol etmektir.



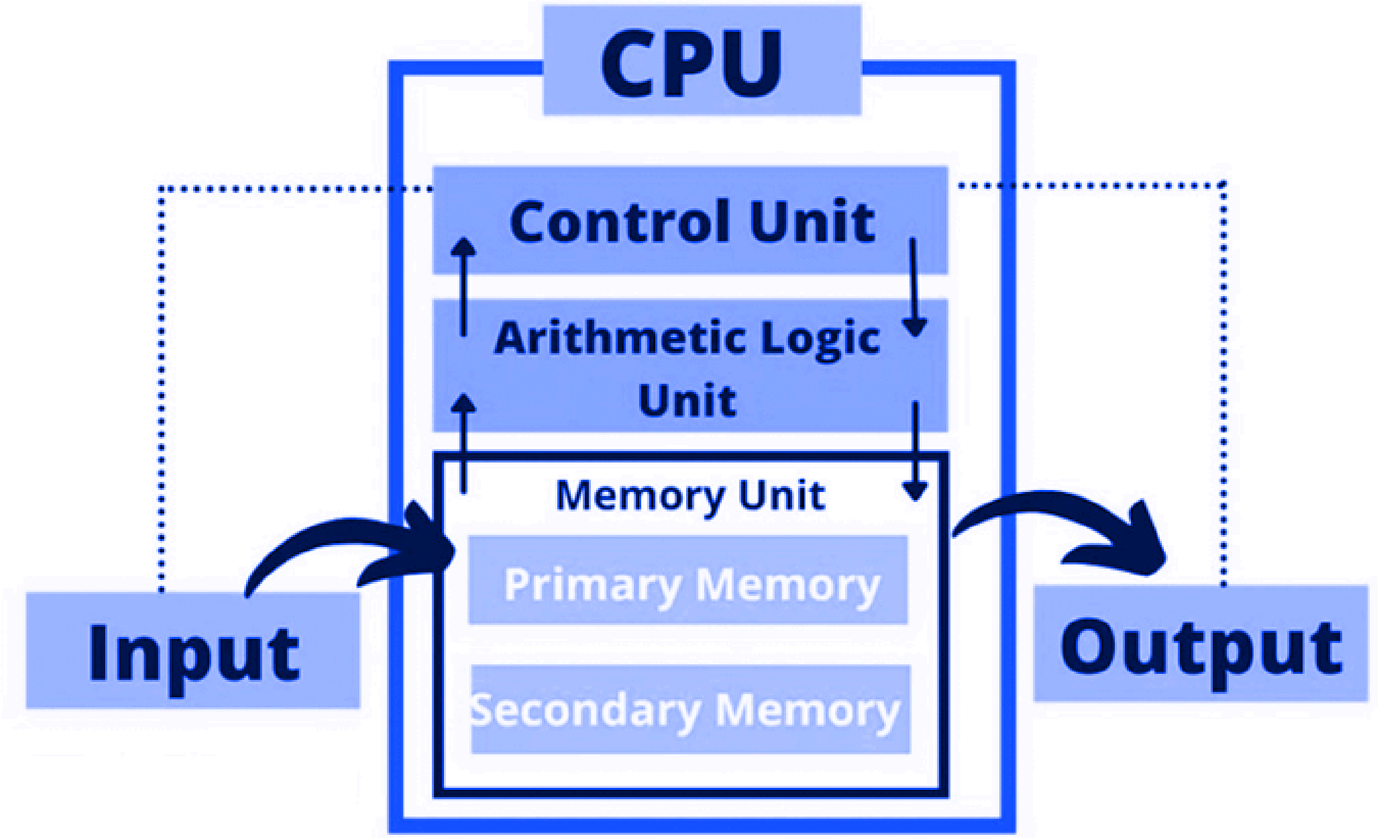
CPU TEMELLERİ

Bir CPU temel olarak üç ana birimden oluşur:

1.ALU(Aritmetik Mantık Birimi)

2.Kontrol Birimi (Control Unit)

3.Register (Kaydedici)


CPU
Control Unit
Arithmetic Logic Unit
Memory Unit
Primary Memory
Secondary Memory
Input
Output

ALU (Aritmetik Mantık Birimi)

- Temel görevi: Veriler üzerinde toplama ve çıkarma gibi temel matematiksel işlemleri gerçekleştiren devreleri barındırır
- Sadece aritmetik değil, aynı zamanda karşılaştırma ve mantıksal işlemleri (AND, OR, XOR vb.) de yürütür
- ALU, işlem yapacağı verileri doğrudan ana bellekten değil, CPU içindeki kaydedici (register) birimlerinden alır.
- İşlem tamamlandığında, üretilen sonuçlar tekrar kaydedici birimlerine aktarılarak depolanır
- Hangi işlemin yapılacağı ve hangi verilerin kullanılacağı konusunda Kontrol Birimi (Control Unit) tarafından yönetilir

Control Unit (Kontrol Birimi)

- Bilgisayarın tüm faaliyetlerini koordine etmekten sorumlu devreleri içeren birimdir.
- Makine içindeki veri akışını ve diğer birimlerin (ALU, Bellek, Kaydediciler) ne zaman çalışacağını yönetir.
- Bellekten gelen komutların getirilmesi (Fetch) ve bu komutların ne anlama geldiğinin çözümlenmesi (Decode) süreçlerini kontrol eder.
- Bir işlem yapılacağı zaman (örneğin toplama), Kontrol Birimi hangi kaydedicilerin veriyi tuttuğunu ALU'ya bildirir ve uygun devreleri aktif hale getirir.
- Verilerin ana bellekten kaydedicilere ne zaman transfer edileceğine ve sonuçların ne zaman belleğe geri yazılacağına karar verir.

REGİSTER(Kaydedici/Yazmaç)

- Kaydediciler, CPU içerisinde yer alan ve ana bellek hücrelerine benzer şekilde çalışan veri depolama hücreleridir
- CPU içinde bilgilerin geçici olarak saklanması için kullanılırlar.
- Ana belleğin (RAM) aksine, kaydediciler doğrudan CPU'nun içinde yer aldıkları için verilere çok daha hızlı erişilmesini sağlarlar.
- Kaydediciler "genel amaçlı" ve "özel amaçlı" olmak üzere ikiye ayrılır; özel amaçlı olanlar programın yürütülmesi sırasında belirli görevleri (örneğin sonraki komutu takip etmek) üstlenirler.
- Genel amaçlı kaydediciler ise CPU tarafından manipüle edilen veriler için geçici tutma yerleri olarak görev yaparlar.

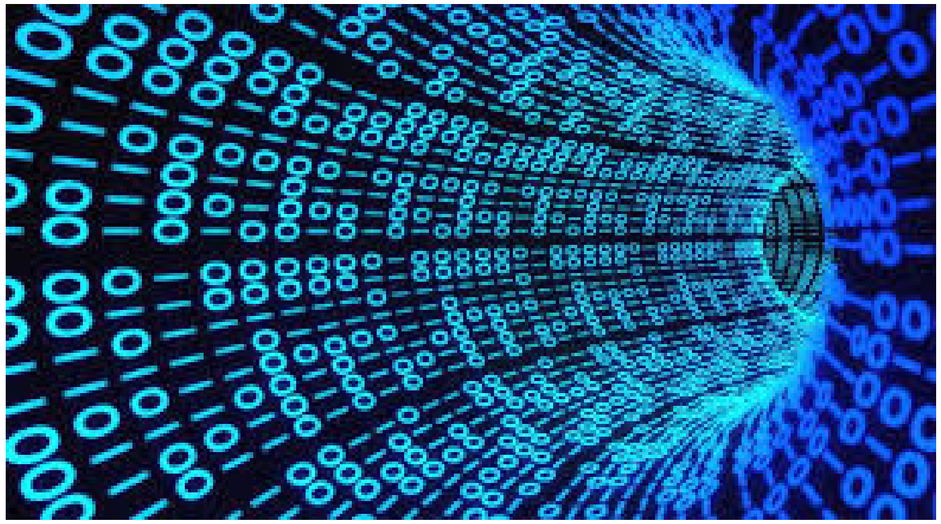
Saklı Program Kavramı (Stored-Program Concept)

- İlk bilgisayarlarda programlar CPU’nun içine gömülüydü ve bu durum bilgisayarların esnek olmasını zorlaştırıyordu.
- Programı değiştirmek için CPU’nun fiziksel olarak yeniden kablolanması gerekiyordu.
- Daha sonra programların da veriler gibi kodlanıp ana bellekte saklanabileceği fikri ortaya çıktı.
- Bu sayede kontrol birimi, komutları bellekten alıp çözüp çalıştırabilir hale geldi.
- Böylece programı değiştirmek için sadece belleğin içeriğini değiştirmek yeterli oldu.
- Bu yaklaşıma Stored-Program Concept (Saklı Program Kavramı) denir ve günümüzde kullanılan bilgisayarların temelini oluşturur.

•••

MAKİNE DİLİ (MACHİNE LANGUAGE)

- Saklı program kavramını uygulamak için işlemcilerin (CPU), bit desenleri (0 ve 1) olarak kodlanmış komutları tanıyacak şekilde tasarlanmasıyla oluşan sisteme makine dili denir.



Bilgisayarlar Komutları Nasıl Anlar?

- Bilgisayarlar komutları, makine dili adı verilen ve ikili (binary) biçimde kodlanmış talimatlar aracılığıyla yorumlar.
- Bu komutlar, CPU tarafından bellekten alınır, çözümlenir ve donanımsal işlemlere dönüştürülerek yürütülür.
- Her komut, işlemcinin anlayabildiği önceden tanımlı bir komut kümesine (instruction set) aittir.

Komut Seti (Instruction Set)

- Bir işlemcinin donanımsal olarak anlayabildiği tüm temel komutların (yönergelerin) toplamıdır.
- İşlemcinin yapabildiği toplama, veri taşıma gibi işlemlerin listesini belirler.
- Yazılımların işlemciye hükmetmesini sağlayan ana arayüzdür.
- Brookshear Örneği: Projemizdeki sanal işlemci, LOAD, STORE, ADD gibi 12 temel komuttan oluşan sade bir sete sahiptir.

İşlemci Tasarım Felsefeleri:RISC ve SISC

- RISC (Sadeleştirilmiş Komut Seti): Sadece en temel ve basit komutları yürüten, yüksek hız ve düşük güç tüketimi hedefleyen mimaridir. Örn: Akıllı telefonlar (ARM).
- CISC (Karmaşık Komut Seti): Tek bir komutla birçok karmaşık işlemi yapabilen, yazılım işlerini kolaylaştıran geniş kapsamlı mimaridir. Örn: Bilgisayarlar (Intel/AMD).
- Brookshear Bağlantısı: Sunumdaki Brookshear makinesi, basit ve sabit yapısıyla RISC felsefesini kullanır.

KOMUT YAPISI

- Op-code (İşlem Kodu): Bir makine komutunun en başında yer alan ve işlemciye "ne yapacağını" (topla, yükle, dur vb.) söyleyen kısımdır.
- Operand (İşlenen): İşlem kodundan sonra gelen ve işlemin "kiminle veya nerede" yapılacağını belirten teknik detaylardır.
- İlişki: Op-code bir fiil (Git), Operand ise o fiilin yönüdür.

Brookshear Makinesi ve Genel Mimari

- Bilgisayar bilimleri eğitimi için tasarlanmış, temsili ve sade bir sanal makine mimarisidir.
- Modern CPU mimarisini anlamak için gerekli olan tüm bileşenleri (bellek, yazmaçlar, komut seti) en basit haliyle sunar.
- 16 Yazmaç (0-F) ve 256 Bellek hücresinden (00-FF) oluşur.
- Her makine komutu tam olarak 2 byte (16 bit) uzunluğundadır.
- Komutların ilk 4 biti Op-code (İşlem Kodu), geri kalan 12 biti ise Operand (İşlenen) alanıdır.

•••

Brookshear Makinesinde Komut Yapısı

- Genel Yapı: Brookshear makinesinde her komut 16 bit uzunluğundadır. Kolay okunabilmesi için bu 16 bit, 4 haneli Hexadecimal (Onaltılık) kod ile gösterilir (Örnek :20A3).
- Bileşenler:

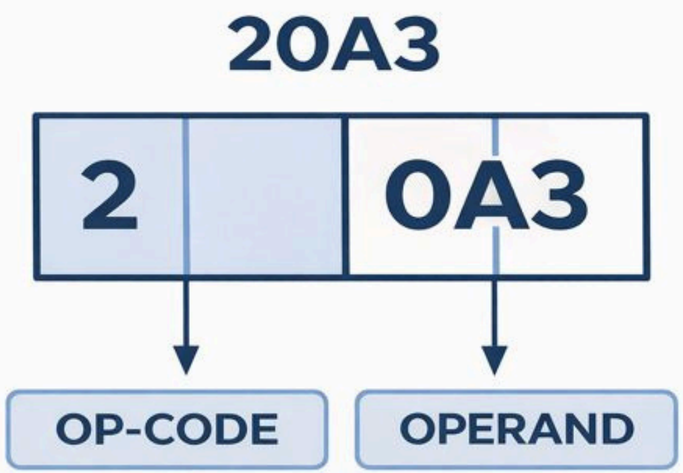
1.Op-code (Operasyon Kodu):

- Komutun ilk hanesidir.
- İşlemciye "ne yapacağını" söyler.
- Örnek: 1 -> "Yükle (LOAD)", 5 -> "Topla (ADD)".

2.Operand (İşlenen / Parametre):

- Komutun geriye kalan 3 hanesidir.
- İşlemciye "hangi veriyi" veya "hangi adresi" kullanacağını söyler.

•••



- Parçalara Ayırma:
- 2 (Op-code): "Kayıtçıya doğrudan veri yükle" talimatıdır.
- 0 (Register): İşlemin hedefi olan 0 numaralı yazmaçtır.
- A3 (Data): Yazmaca yüklenecek olan gerçek bit desenidir (onaltılık değer).
- Sonuç: "0 numaralı yazmaca A3 değerini koy.".

12 Temel Komuta Giriş (Instruction Set)

- Kapsam: Bu sanal makine sadece 12 temel komut (1'den C'ye kadar Op-codelar) ile çalışır.
- Esneklik: Bu 12 komut, karmaşık algoritmaları ve matematiksel işlemleri gerçekleştirmek için yeterli bir "Instruction Repertoire" oluşturur.

Gösterim Sembolleri:

- R, S, T: Belirli yazmaçları temsil eden onaltılık hanelerdir.
- X, Y (veya XY): Bellek adreslerini veya sabit değerleri temsil eden hanelerdir.

•••

Veri Aktarım Komutları (Op-code: 1, 2, 3, 4)

- 1 RXY (LOAD): XY adresindeki bellek hücresinin içeriğini R yazmacına yükler. (Örn: 14A3 → A3 hücresini Yazmaç 4'e yükle).
- 2 RXY (LOAD): Doğrudan XY sabit değerini R yazmacına yükler. (Örn: 20A3 → Yazmaç 0'a A3 değerini koy).
- 3 RXY (STORE): R yazmacındaki veriyi XY adresindeki bellek hücresine saklar. (Örn: 35B1 → Yazmaç 5'i B1 hücresine yaz).
- 4 0RS (MOVE): R yazmacındaki veriyi S yazmacına kopyalar. (Örn: 40A4 → Yazmaç A'yı Yazmaç 4'e kopyala).

•••

Aritmetik ve Mantık Komutları (Op-code: 5, 6, 7, 8, 9, A)

- 5 RST (ADD): S ve T yazmaçlarını "Two's Complement" (İkiye Tümleyen) sisteminde toplar, sonucu R'ye yazar.
- 6 RST (ADD): S ve T yazmaçlarını "Floating-Point" (Kayan Noktalı) sistemde toplar, sonucu R'ye yazar.
- 7/8/9 RST (Mantıksal): S ve T yazmaçları arasında sırasıyla OR, AND ve XOR işlemlerini yapar.
- A R0X (ROTATE): R yazmacındaki bitleri dairesel olarak X kez sağa döndürür. (Örn: A403).

•••

Kontrol ve Dallanma Komutları (Op-code: B, C)

- B RXY (JUMP): Eğer R yazmacındaki veri Yazmaç 0 ile aynı ise, program akışını XY adresindeki komuta atlatır
  - Bu komut "koşullu dallanma" sağlar ve programların karar vermesine olanak tanır.

- C 000 (HALT): Programın yürütülmesini tamamen durdurur. (Örn: C000).

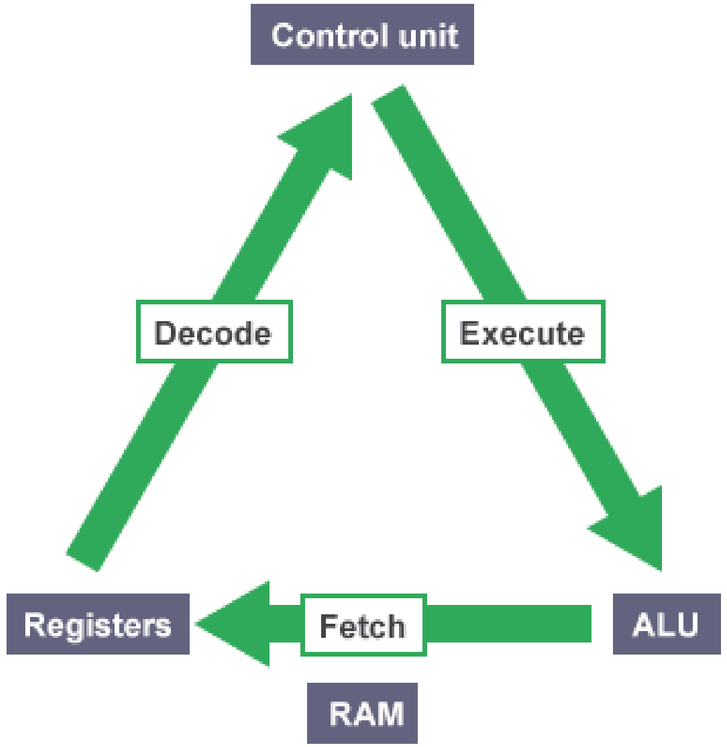
•••

A simple machine language (from Computer science: An overview, by G. Brookshear and D. Brylow, 12th edition)

Form of instruction	Meaning of the instruction
1RXY	LOAD register R with the bit pattern found in the memory cell whose address is XY
2RXY	LOAD register R with the bit pattern XY
3RXY	STORE the bit pattern found in register R in the memory cell whose address is XY
40RS	MOVE the bit pattern found in register R to register S
5RST	ADD the bit patterns in registers S and T and leave the result in register R (the patterns in S and T are treated as 2's complement representations)
6RST	ADD the bit patterns in registers S and T as though they represented values in floating-point notation and leave the floating-point result in register R
7RST	OR the bit patterns in registers S and T and place the result in register R
8RST	AND the bit patterns in registers S and T and place the result in register R
9RST	XOR the bit patterns in registers S and T and place the result in register R
AR0X	ROTATE the bit pattern in register R one bit to the right X times. Each time place the bit that started at the low-order end at the high-order end
BRXY	JUMP to the instruction located in memory cell at address XY if the bit pattern in register R equals the bit pattern in register 0
C000	HALT the execution

Programın Yürütülmesi: Makine Döngüsü

Bir bilgisayarın, bellekteki komutları sırayla alıp işleme sürecine "Makine Döngüsü" (Machine Cycle) denir.
Bilgisayar çalıştığı sürece bu döngü (Getir -> Çöz -> Yürüt) durmaksızın tekrarlanır.



Bu süreçte CPU içindeki iki özel yazmaç (Register) kritik rol oynar:

1. Program Sayacı (Program Counter - PC): Sıradaki komutun adresini tutar.
2. Komut Yazmacı (Instruction Register - IR): O an işlenen komutun kendisini tutar.

GETİRME (FETCH)

- İşlemci, Program Sayacı'nda (PC) yazan adrese gider ve o adresteki komutu ana bellekten alır.
- Bellekten getirilen bu komut, işlenmek üzere Komut Yazmacı'na (IR) yerleştirilir.
- Komut getirildikten sonra, PC otomatik olarak bir sonraki komutun adresine güncellenir (Brookshear'da komutlar 2 byte olduğu için PC değeri 2 artırılır).

ÇÖZME (DECODE)

- Komut Yazmacı'ndaki (IR) bit desenleri CPU'nun kontrol birimi tarafından analiz edilir.
- Komutun Op-code (ne yapılacak?) ve Operand (kiminle yapılacak?) kısımları birbirinden ayrılır.
- İşlemci, komutu yerine getirmek için gerekli olan donanımsal yolları ve devreleri (ALU gibi) hazırlar.

Yürütme (EXECUTE)

- Çözülen komutun gerektirdiği işlem fiziksel olarak gerçekleştirilir.

Örnekler:

- Eğer komut bir ADD ise, iki sayı toplanır.
- Eğer bir LOAD ise, bellekten veri çekilir.
- İşlem tamamlandığında döngü başa döner ve yeni komut "Fetch" edilir.

Python Projesi Tanıtımı

- Bu Python projesi, Brookshear makine dilinde yazılmış 4 haneli hexadecimal komutların ne işe yaradığını Türkçe olarak açıklamak amacıyla geliştirilmiştir.

- Program kullanıcıdan bir HEX komut alır;

```
kod = input("4 haneli HEX kodu giriniz (Cikmak icin x): ").upper()
```

- Bu komutu Op-code, Register ve Operand olarak parçalara ayırır ;

```
# Kod parcalama islemleri
#Örnek: 2F0A --> Opcode:2, Kaydedici:F, Operand:0A
opcode = kod[0]
register = kod[1]
operand = kod[2] + kod[3]
```

Python Projesi Tanıtımı

- Ve ilgili komutun ne yaptığı ekrana yazdırır

```
    elif opcode == "2":
        print("Degeri", operand, "olan bit desenini,",
              register, "numarali kaydediciye (Register) yukle.")
```

- Örnek: Kullanıcı 20A3 komutunu girdiğinde, çıktı:

```
 === Brookshear Makine Dili Yorumlayici===
4 haneli HEX kodu giriniz (Cikmak icin x): 20A3
Degeri A3 olan bit desenini, 0 numarali kaydediciye (Register) yukle.
4 haneli HEX kodu giriniz (Cikmak icin x):
```

Kaynakça

1. *Ana Kaynak: Brookshear, J. G., & Brylow, D. (2014). Computer Science: An Overview (12th Edition). Pearson. Bölüm Referansı: "Chapter 2: Data Manipulation", Section 2.2 (Machine Language).*

2. *Ek Kaynak (Appendix): "Appendix C: A Simple Machine Language" (Brookshear Machine Architecture and Instruction Set).*

3. Görsel Materyaller: Computer Science: An Overview kitabındaki şablon.

4.https://tr.wikipedia.org/wiki/Makine_dili
https://prezi.com/p/qox3ahplwoan/makine-dili-ve-brookshear-mimarisi/