# Bit, Byte ve Integer

BIL-304: Bilgisayar Mimarisi

**Dersi veren öğretim üyesi:**

Dr. Öğr. Üyesi Fatih Gökçe

Ders kitabına ait sunum dosyalarından adapte edilmiştir: http://csapp.cs.cmu.edu/

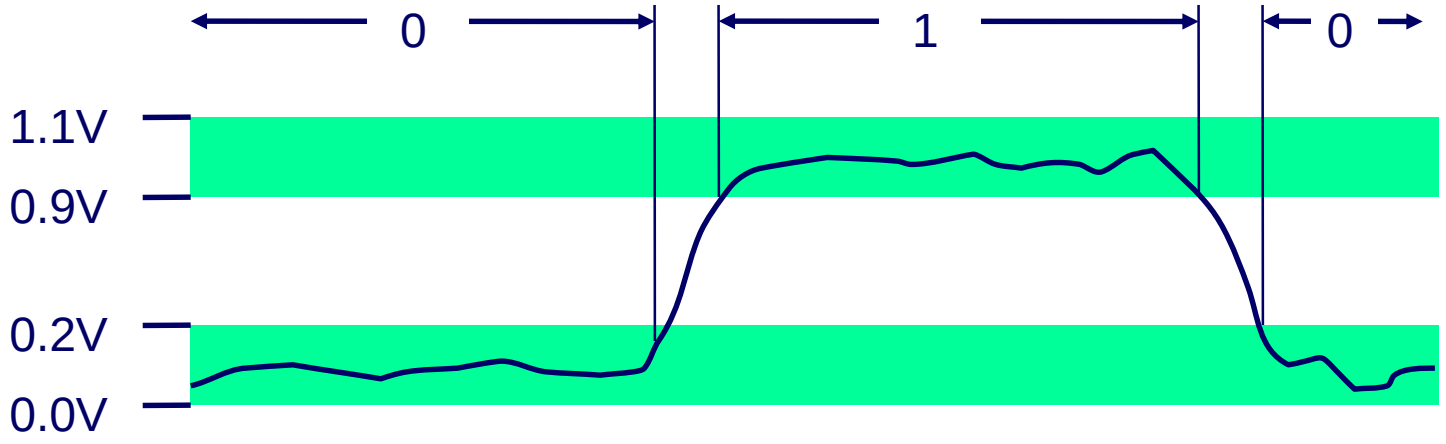
Adapted from slides of the textbook: http://csapp.cs.cmu.edu/

# Bit, Byte ve Integer

- **Bilginin bitler ile ifade edilmesi**
- **Bit düzeyinde manipülasyonlar**
- **Integer**
  - unsigned ve signed
  - Dönüştürme (conversion), tip dönüşümü (casting)
  - Genişletme (expanding), kırpma (truncating)
  - Toplama, işaret değiştirme (negation), çarpma, kaydırma
- **Bilginin hafızada saklanma şekilleri, pointer'lar, karakter dizileri (string'ler)**

# Herşey bit ile ifade edilir

- **Her bit 0 veya 1'dir.**
- **Bit gruplarının değişik şekillerde kodlanması/anlamlandırılması ile**
  - Bilgisayarlar ne yapacaklarına karar verirler (komutlar, opcode lar..)
  - … ve sayıları, karakter dizilerini, vs ifade edip manipüle ederler.
- **Neden bit? Elektronik sistem olarak uygulama kolaylığı**
  - İki durumlu elektronik elemanlarla kaydetmek kolaydır.
  - Gürültü altında bile analog sinyallere göre daha güvenli iletilebilirler.



# Byte Değerlerinin Kodlanması

- **Byte = 8 bit**
  - Binary $00000000_2$ den $11111111_2$ ye kadar.
  - Onluk tabanda: $0_{10}$ - $255_{10}$
  - Onaltılık tabanda: $00_{16}$ - $FF_{16}$
    - 16 tabanında sayıların ifade edilmesi
    - '0' dan '9' a rakamlar ve 'A' dan 'F' e karakterler
    - C dilinde $FA1D37B_{16}$ şu şekillerde yazılır:
      - 0xFA1D37B
      - 0xfa1d37b

| Onaltılık | Onluk | Binary |
|---|---|---|
| 0 | 0 | 0000 |
| 1 | 1 | 0001 |
| 2 | 2 | 0010 |
| 3 | 3 | 0011 |
| 4 | 4 | 0100 |
| 5 | 5 | 0101 |
| 6 | 6 | 0110 |
| 7 | 7 | 0111 |
| 8 | 8 | 1000 |
| 9 | 9 | 1001 |
| A | 10 | 1010 |
| B | 11 | 1011 |
| C | 12 | 1100 |
| D | 13 | 1101 |
| E | 14 | 1110 |
| F | 15 | 1111 |

# C'de veri tiplerinin tipik boyutları

| C Veri Tipi | 32-bit program | 64-bit program |
|---|---|---|
| `char` | 1 | 1 |
| `short` | 2 | 2 |
| `int` | 4 | 4 |
| `long` | 4 | 8 |
| `int32_t` | 4 | 4 |
| `Int64_t` | 8 | 8 |
| `float` | 4 | 4 |
| `double` | 8 | 8 |
| `char *` | 4 | 8 |

# Bit, Byte ve Integer

- Bilginin bitler ile ifade edilmesi
- **Bit düzeyinde manipülasyonlar**
- Integer
  - unsigned ve signed
  - Dönüştürme (conversion), tip dönüşümü (casting)
  - Genişletme (expanding), kırpma (truncating)
  - Toplama, işaret değiştirme (negation), çarpma, kaydırma
- Bilginin hafızada saklanma şekilleri, pointer'lar, karakter dizileri (string'ler)

# Boole Cebri

- **George Boole tarafından 19. yüzyılda geliştirilmiştir.**
  - Lojiğin cebirsel ifadesidir
    - "Doğru" 1 ile "Yanlış" 0 ile kodlanır.

## Ve (And)

- **Hem A=1 hem de B=1 ise A&B = 1**

| & | 0 | 1 |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |

## Veya (Or)

- **A=1 veya B=1 ise A|B = 1**

| \| | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 1 |

## Değil (Not)

- **A=0 ise ~A = 1**

| ~ | |
|---|---|
| 0 | 1 |
| 1 | 0 |

## Özel-veya (Exclusive-Or, Xor)

- **A=1 veya B=1 ise fakat ikisi birden 1 değilse A^B = 1**

| ^ | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 0 |

# Genel Boole Cebri

- **Bit vektörleri (bit dizileri) üzerinde işlemler**
  - İşlemler bit düzeyinde karşılıklı bitler arasında uygulanır

```
  01101001        01101001        01101001
& 01010101      | 01010101      ^ 01010101      ~ 01010101
----------      ----------      ----------      ----------
  01000001        01111101        00111100        10101010
```

- **Boole cebrinin tüm özellikleri geçerlidir**

# Küme şeklinde gösterim ve Manipülasyonlar

- **Gösterim**
  - w genişliğinde bir A bit dizisi {0, …, w–1} kümesinin bir alt kümesidir
  - Eğer j ∈ A ise $a_j$ = 1'dir. $a_j$, A'daki herbir biti göstermektedir.

    - 01101001 { 0, 3, 5, 6 } → A
    - *76543210*

    - 01010101 { 0, 2, 4, 6 } → B
    - *76543210*

- **A ve B kümeleriyle işlemler**

| | | | |
|---|---|---|---|
| & | Kesişim | 01000001 | { 0, 6 } |
| \| | Birleşim | 01111101 | { 0, 2, 3, 4, 5, 6 } |
| ^ | Simetrik Fark (XOR) | 00111100 | { 2, 3, 4, 5 } |
| ~ | Tümleyen | 10101010 | { 1, 3, 5, 7 } |

# C'de Bit-Düzeyinde İşlemler

- **C'de &, |, ~, ^ işlemleri bulunmaktadır.**
  - Tam sayıların sonlu bir alt kümesini gösteren tüm veri tiplerine ("integral" veri tipi) uygulanabilirler.
    - long, int, short, char, unsigned
  - Argümanları bir vektörü olarak görürler
  - Argümanlar bit-düzeyinde işleme tabi tutulurlar
- **Örnekler (Char veri tipi için)**
  - ~0x41 → 0xBE
    - $\sim 01000001_2 \rightarrow 10111110_2$
  - ~0x00 → 0xFF
    - $\sim 00000000_2 \rightarrow 11111111_2$
  - 0x69 & 0x55 → 0x41
    - $01101001_2 \;\&\; 01010101_2 \rightarrow 01000001_2$
  - 0x69 | 0x55 → 0x7D
    - $01101001_2 \mid 01010101_2 \rightarrow 01111101_2$

# C'de Lojik (Mantıksal) İşlemler

- **Bit-Düzeyinde işlemlerden farklıdırlar**
  - Semboller → VE: &&, VEYA: ||, DEĞİL: !
    - 0 → "Yanlış"
    - Sıfırdan farklı değer → "Doğru"
    - Ya 0 (Yanlış) ya da 1 (Doğru) değeri dönerler
    - Lojik ifadenin değeri birinci argümana bakılarak belirlenebiliyorsa ikinci argüman değerlendirilmez.
- **Örnekler (char veri tipi. İşlemlerin bit-düzeyindeki karşılıklarından farklı sonuçlar çıkabildiğine dikkat ediniz!)**
  - !0x41 → 0x00
  - !0x00 → 0x01
  - !!0x41 → 0x01
  - 0x69 && 0x55 → 0x01
  - 0x69 || 0x55 → 0x01 (0x69 Doğru olduğu için VEYA operatöründen sonra gelen ikinci argümanı değerlendirmeye gerek kalmaz)
  - p && *p (Null pointer erişimi hatasını önler, p=0 ise p bir Null pointer'dir, operatör VE olduğu için *p yi değerlendirmeden sonucun 0 olduğu sonucuna varılır.)
  - a && 5/a (a=0 ise ikinci argüman işletilmeden 0 döner; hiçbir zaman sıfıra bölme hatası ile karşılaşılmaz.)

# C'de Lojik (Mantıksal) İşlemler

- **Bit-Düzeyinde işlemlerden farklıdırlar**
  - Semboller → VE: &&, VEYA: ||, DEĞİL: !
    - 0 → "Yanlış"
    - Sıfırdan farklı değer → "Doğru"
    - Ya 0 (Yanlış) ya da 1 (Doğru) değeri dönerler
    - Lojik ifadenin değeri birinci argümana bakılarak belirlenebiliyorsa ikinci argüman değerlendirilmez.
- **Örnekler (char veri tipi. İşlemlerin bit-düzeyindeki karşılıklarından farklı sonuçlar çıkabildiğine dikkat ediniz!)**
  - !0x41 → 0x00
  - !0x00 → 0x01
  - !!0x41 → 0x01
  - 0x69 && 0x55 → 0x01
  - 0x69 || 0x55 → 0x01 (0x69 Doğru olduğu için VEYA operatöründen sonra gelen ikinci argümanı değerlendirmeye gerek kalmaz)
  - p && *p (Null pointer erişimi hatasını önler, p=0 ise p bir Null pointer'dir, operatör VE olduğu için *p yi değerlendirmeden sonucun 0 olduğu sonucuna varılır.)
  - a && 5/a (a=0 ise ikinci argüman işletilmeden 0 döner; hiçbir zaman sıfıra bölme hatası ile karşılaşılmaz.)

# C'de Lojik (Mantıksal) İşlemler

- **Bit-Düzeyinde işlemlerden farklıdırlar**
  - Semboller → VE: &&, VEYA: ||, DEĞİL: !
    - 0 → "Yanlış"
    - Sıfırdan farklı değer → "
    - Ya 0 (Yanlış) ya
    - Lojik ifadenin d an değerlendirilme
- **Örnekler (char ver sonuçlar çıkabildiğ**
  - !0x41 → 0x00
  - !0x00 → 0x01
  - !!0x41 → 0x01
  - 0x69 && 0x55 → 0x01
  - 0x69 || 0x55 → 0x01 (0x69 Doğru olduğu için VEYA operatöründen sonra gelen ikinci argümanı değerlendirmeye gerek kalmaz)
  - p && *p (Null pointer erişimi hatasını önler, p=0 ise p bir Null pointer'dir, operatör VE olduğu için *p yi değerlendirmeden sonucun 0 olduğu sonucuna varılır.)
  - a && 5/a (a=0 ise ikinci argüman işletilmeden 0 döner; hiçbir zaman sıfıra bölme hatası ile karşılaşılmaz.)

**DİKKAT!**
**&& Mantıksal, & Bit-düzeyinde VE**
**|| Mantıksal, | Bit-düzeyinde VEYA**

# Kaydırma (Shift) İşlemleri

- **Sola Kaydırma: x << y**
  - x bit vektörünü y kadar sola kaydırır.
    - Soldaki ekstra bitler atılır.
    - Sağdaki kayan bitlerin yeri 0 ile doldurulur.
- **Sağa kaydırma: x >> y**
  - x bit vektörünü y kadar sağa kaydırır.
    - Sağdaki ekstra bitler atılır.
  - İki tipi vardır: Lojik ve Aritmetik
  - Lojik kaydırma
    - Soldaki kayan bitlerin yeri 0 ile doldurulur.
  - Aritmetik kaydırma
    - Soldaki kayan bitlerin yeri işaret biti ile doldurulur.
- **Tanımlanmamış durumlar**
  - Kaydırma miktarı 0'dan küçük ise veya word size'a eşit ya da word size'dan büyük ise.

| **Argüman** x | 01100010 |
|---|---|
| << 3 | 00010000 |
| **Lojik** >> 2 | 00011000 |
| **Arit.** >> 2 | 00011000 |

| **Argüman** x | 10100010 |
|---|---|
| << 3 | 00010000 |
| **Lojik** >> 2 | 00101000 |
| **Arit.** >> 2 | 11101000 |

İki tip sağa kaydırma olmasına rağmen C'de tek bir sağa kaydırma operatörü vardır, >>. Birçok C compiler'ı sağa kaydırmanın tipini integer'ın türüne göre belirler; sıklıkla signed integer'lar aritmetik kaydırmaya göre, unsigned integer'lar da lojik kaydırmaya göre sağa kaydırılır.