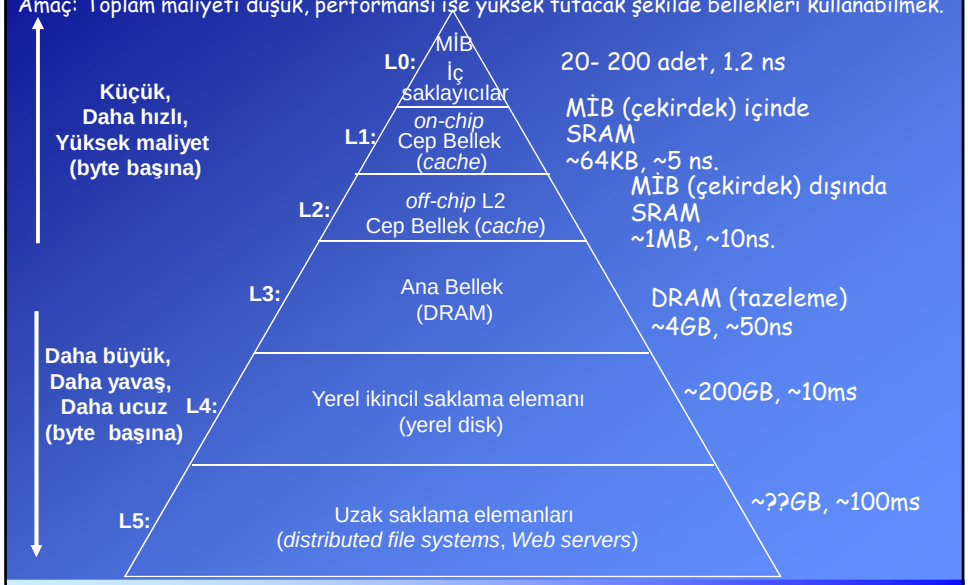
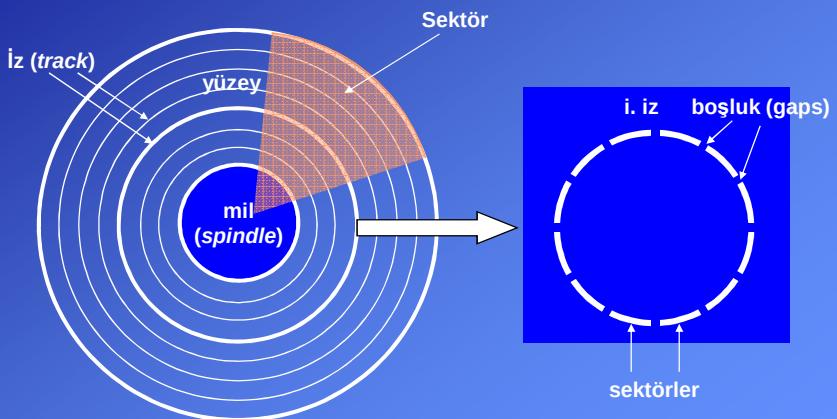
Bellek Organizasyonu

Elimizde farklı hız, boyut ve fiyatlarda bellekler var.
Amaç: Toplam maliyeti düşük, performansı ise yüksek tutacak şekilde bellekleri kullanabilmek.

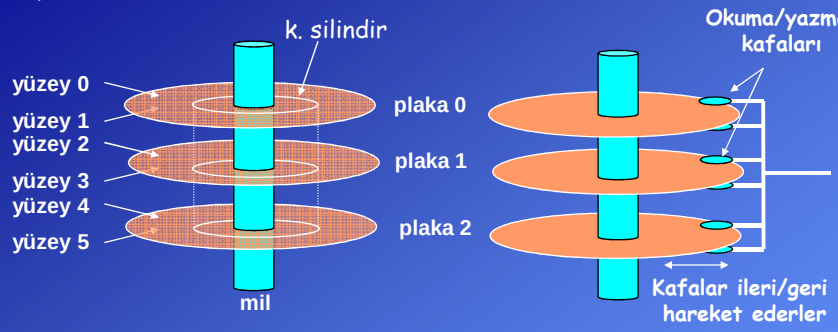


DİSK:

- Her plakada iki yüzey.
- Her yüzeyde eş merkezli halkalar (izler) bulunur.
- Her izde boşluklarla (*gaps*) ayrılan sektörler bulunur.



Aynı hizadaki izler bir silindir oluşturur.



Kapasite = (byte/sektör) x (ort. sektör/iz) x (iz/yüzey) x (yüzey/plaka) x (plaka/disk)

Örnek:

512 byte/sektör
300 sektör/iz (ortalama)
20,000 iz/yüzey
2 yüzey/plaka
5 plaka/disk

Kapasite = 512 x 300 x 20000 x 2 x 5
= 30,720,000,000 byte
≈ 28.6 GB

Disk Erişim Süresi

Bir diskin ortalama erişim süresi üç bileşenden oluşur:

Erişim süresi (Ta) = Konumlanma süresi (Ts) + Dönüş gecikmesi (Tr) + Aktarım Süresi(Tt)

- **Ortalama konumlanma süresi (*Seek time*) Ts:**

Okuma/yazma kafasının ilgili ize konumlanması için geçen süre. Yaklaşık 9ms (3-15ms)

- **Ortalama dönüş gecikmesi (*Rotational latency*) Tr:**

Okuma/yazma kafasının, iz içinde ilgili sektörün başına konumlanması için geçen süre.
Kafa diskte ilgili izin üstüne konumlandıktan sonra gerekli olan sektörün gelmesi için plakanın dönmesini bekler.
Bu bekleme süresi ortalama olarak diskin bir turunu tamamlaması için gerekli olan sürenin yarısı kadardır:

$$T_r = \frac{1}{2} r$$ r: Diskin bir tur dönüş süresi (saniye)

Genellikle disklerin dönüş hızları tur/dakika (RPM: *Revolution per minute*) olarak verilir. Buna göre dönüş gecikmesi saniye cinsinden aşağıdaki gibi hesaplanabilir:

$$T_r = \frac{1}{2} \frac{60}{RPM}$$

Örnek: 7200 RPM disk bir turunu 8.3 ms'de tamamlar. Buna göre ortalama dönüş gecikmesi yaklaşık 4ms'dir.
10000 rpm: 3ms, 15000 rpm: 2ms.

•Aktarım Süresi (*Transfer time*) (Tt)

İki farklı şekilde ifade edilebilir: Bir sektörü aktarmak için geçen süre ya da belli miktarda byte aktarmak için geçen süre.

Bir sektörü okumak için geçen süre (Tts).

$$T_{ts} = \frac{1}{ort.sektör/iz}\frac{60}{RPM} \text{ [saniye]}$$

Örnek: Bir diskin dönüş hızı 7200 RPM ise ve bir izinde ortalama olarak 400 sektör varsa bir sektörlük aktarım hızı aşağıdaki gibi hesaplanır:

Tts = 60/7200 RPM x 1/400 sektör/iz x 1000 ms/sec = 0.02 ms

Veri aktarım süresi (Ttb).

b: Aktarılacak byte sayısı, N: Bir izdeki byte sayısı

$$T_{tb} = \frac{b}{N}\frac{60}{RPM} \text{ [saniye]}$$

Örnek:

-Disk dönüş hızı = 7200 RPM
-Ortalama konumlanma süresi = 9 ms.
-Bir izdeki ortalama sektör sayısı = 400.

Buna göre:

-Dönüş gecikmesi = 1/2 x (60 s/7200 RPM) x 1000 ms/s = 4 ms.
-Aktarım süresi = 60/7200 RPM x 1/400 sektör/iz x 1000 ms/sec = 0.02 ms
-Erişim süresi = 9 ms + 4 ms + 0.02 ms

Sonuç:

-Erişim süresinin belirleyici bileşenleri konumlanma süresi ve dönüş gecikmesidir.
-Sektörün ilk bitine ulaşmak zaman kaybettirir.
-Disk SRAM'dan 40,000 kat daha yavaştır,
-Disk DRAM'dan 2,500 kat daha yavaştır.
-Veri aktarım hızları: Standart masaüstü bir bilgisayardaki 7200 RPM SATA disk
  - Arabirim tampon belleği – fiziksel plakalar arası: 1030 Mbit/s
  - Bilgisayar belleği – arabirim tampon belleği arası: 300 Mbyte/s

RAID: (*Redundant Array of Independent/Inexpensive Disks*)
Fazlalıklı Bağımsız/Ucuz Diskler Dizisi

Veriler paralel çalışan birden çok diske dağıtılır.

Amaç:

Performansı ve güvenirliği arttırmak.

Paralel ve bağımsız diskler performansı arttırır. Fazlalık bilgi, hataları sezmek ve düzeltmek için kullanılır.

Düzeyler:

RAID 1 – RAID 7: 7 değişik düzey ve bunların bileşiminden oluşan bileşik düzeyler var

Ayrıca tam bir RAID düzeyi olmamakla birlikte konuyu anlamak için kullanılan RAID 0

En çok RAID 3 ve 5 kullanılır.

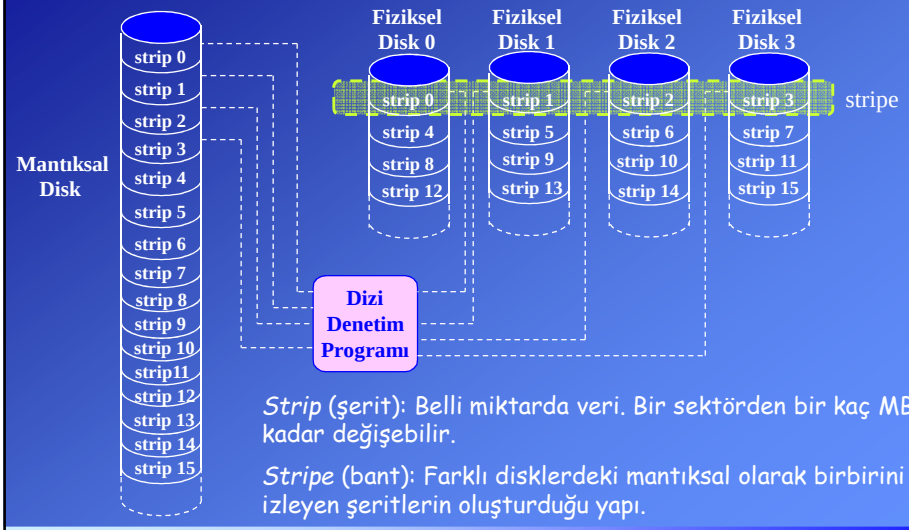
Ortak özellikler:

1. Birden fazla fiziksel disk vardır. İşletim sistemleri bunları bir bütün, tek bir mantıksal disk olarak görür (gösterir).
2. Mantıksal olarak peş peşe gelen veriler belli büyüklükteki bloklar (şerit –"*strip*") halinde farklı fiziksel disklere paralel olarak yerleştirilirler.
3. Fazlalık olarak eşlik bilgileri (*parity*) yerleştirilir. Disklerden biri fiziksel olarak bozulduğunda bu diskteki bilgi tekrar oluşturulabilir.

RAID 0

Fazlalık ve eşlik biti yok. Hatalar düzeltilemez. Tam bir RAID sistemi değildir.

Veriler paralel erişilebilen fiziksel disklere dağıtılır, performans artımı sağlanır.



Strip (şerit): Belli miktarda veri. Bir sektörden bir kaç MB'a kadar değişebilir.

Stripe (bant): Farklı disklerdeki mantıksal olarak birbirini izleyen şeritlerin oluşturduğu yapı.

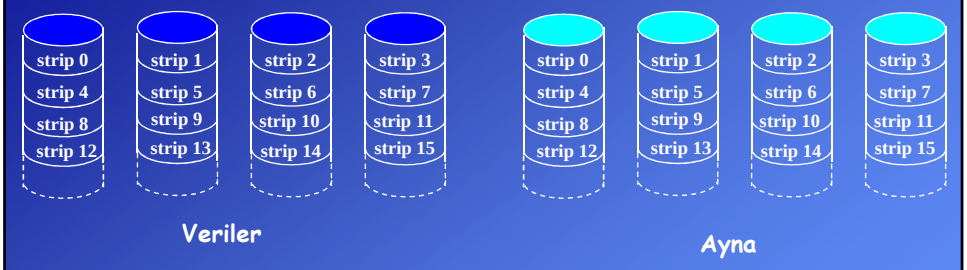
Şerit (*strip*) boylarının performansa etkisi:

- Eğer mantıksal olarak **peş peşe gelen büyük bloklara erişiliyorsa** paralel erişim sağlayabilmek için mantıksal olarak birbirini izleyen verilerin mümkün olduğu kadar farklı fiziksel disklere dağıtılması istenir.
  Bu durumda **küçük şeritler** performansı arttırır.

- Eğer **sık G/Ç istekleri varsa ve küçük bloklara erişiliyorsa** (örneğin kısa ve sık veri tabanı sorguları) erişimlerin farklı disklere olması için **büyük şeritler** tercih edilir.

RAID 1

Veriler aynalanır (*mirroring*). Her veri iki ayrı diske yazılır.



İki disk paralel (aynı anda) okunmaya başlanır. Daha hızlı olandan veri alınır.

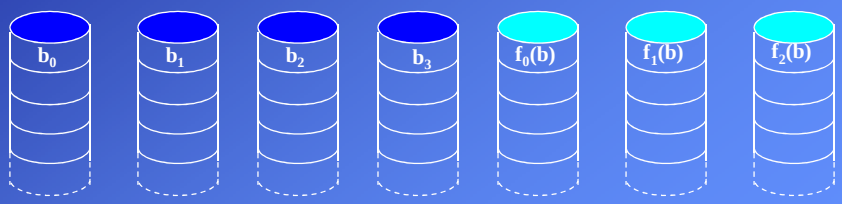
Yazarken veri iki diske de paralel yazılır. Bu durumda daha yavaş olan disk beklenir.

Bir disk bozulduğunda veri diğerinden alınır. Fiziksel olarak hangi diskin bozulduğu bellidir.

Fazla disk kullandığı için pahalı bir yöntemdir.

RAID 2

- Hata sezme/düzeltme için eşlik bitleri eklenir.
- Hata sezme için Hamming kodu kullanılır.
- Hamming kodlaması (sonraki bölümde açıklanacak) belleklerde ve iletişimde hata sezme/düzeltme için kullanılır.
- RAID 2'de diskler senkron çalışır, tüm disklerdeki kafalar aynı yere konumlanır.
- Küçük "strip"ler (1 sözcük) kullanılır. Sürekli ve büyük blok erişimlerinde avantajlı.
- Aşağıdaki şekilde 4 bitlik veriye hata sezme için 3 bitlik eşlik eklenmiştir.
- Disklerin fiziksel olarak bozulduğunu anlamak mümkün olduğu için disklerde Hamming kodlaması gereksiz yere karmaşıklığı arttırmakta ve maliyeti yükseltmektedir.



RAID 3

- RAID 2'de olduğu gibi diskler senkron çalışır, tüm disklerdeki kafalar aynı yere konumlanır.
- Küçük "strip"ler kullanılır. Sürekli ve büyük blok erişimlerinde avantajlı.
- Hata sezme/düzeltme için daha basit bir kodlama kullanılır ve tek eşlik biti eklenir.
- Eşlik (*parity*) biti, veri bitleri "ya da"lanarak belirlenir. X0-X3 veri sözcükleri, X4 ise eşlik sözcüğü olmak üzere eşlik bitleri aşağıdaki gibi hesaplanır:

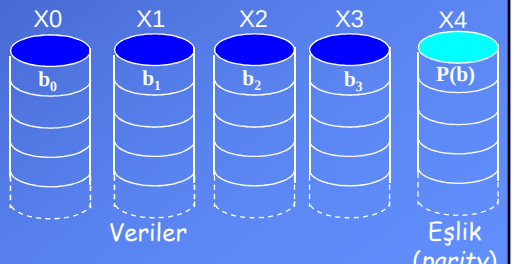
$X4(i) = X0(i) \oplus X1(i) \oplus X2(i) \oplus X3(i)$; Böylece 1'lerin toplam sayısı çift olur.

- Normalde bu eşlik yöntemi sadece tek sayıdaki hataları sezebilir ama düzeltemez. Ancak fiziksel olarak hangi diskin bozulduğu belli ise eşlik bilgilerinden yararlanılarak o diskteki bilgiler yeniden oluşturulabilir.

Örneğin 1 numaralı disk bozulursa:

$X1(i) = X0(i) \oplus X2(i) \oplus X3(i) \oplus X4(i)$

- Her yazmada eşlik diskine de erişmek gerekir.
- Diskler senkron çalıştığı için bağımsız olarak disklerin farklı bölgelerine erişmek mümkün değildir.
- Sık, bağımsız erişimlerde performans düşer.



RAID 4

- Diskler bağımsız çalışır (senkron değil).
- Büyük "strip"ler (blok) kullanılır. Sık ve bağımsız okuma erişimlerinde avantajlı.
- Hata sezme/düzeltme için eşlik biti eklenir.
- Tek bir eşlik diski vardır.
- Okumada eşlik diski okunmaz ancak yazma işlemlerinde disklerin farklı yerlerine aynı anda yazmak mümkün olmaz çünkü eşlik diski tektir, beklemek gerekir.
- Yazma cezası (*write panalty*): Bir diske yazılırsa, örneğin disk 1, eşlik hesaplanır:

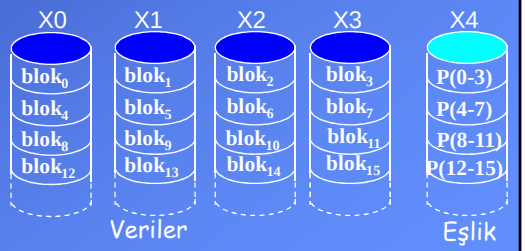
$X4'(i) = X0(i) \oplus X'1(i) \oplus X2(i) \oplus X3(i)$

Bu durumda 3 diskten okumak (X0,X2,X3), 2 diske yazmak (X4,X1) gerekir.

Biraz kolaylaştırmak için sağ tarafa $\oplus X1(i) \oplus X1(i)$ eklenir.

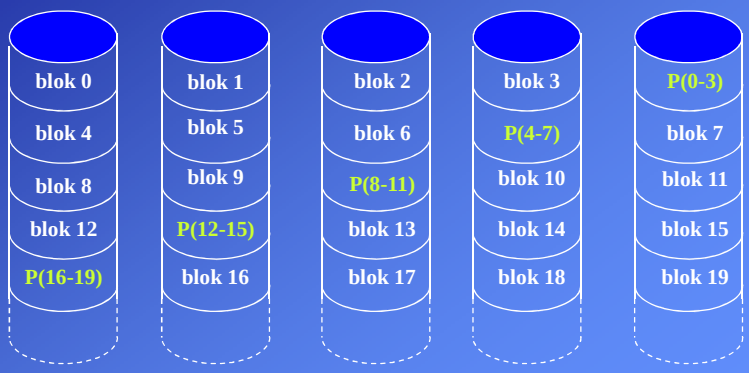
$X4'(i) = X4(i) \oplus X'1(i) \oplus X1(i)$ olur.

Bu durumda eşlik hesabı yapabilmek için iki okuma iki yazma gereklidir. Değiştirilecek verinin ve eşlik bitlerinin eski değerleri okunur yeni değerleri yazılır.



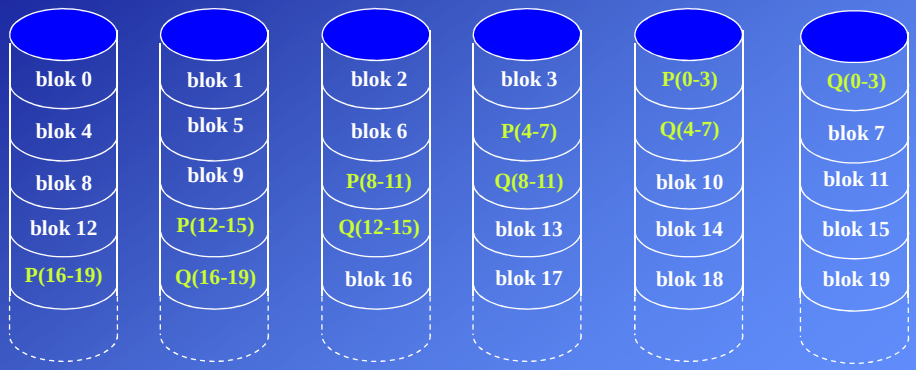
RAID 5

- RAID 4' e benzer. Diskler bağımsız çalışır (senkron değil).
- Büyük "strip"ler (blok) kullanılır. Sık ve bağımsız okuma erişimlerinde avantajlı.
- Hata sezme/düzeltme için eşlik biti eklenir.
- RAID 4'ten farklı olarak eşlik bilgileri disklere dağıtılır. Böylece her yazma işleminde aynı eşlik diskinin beklenmesi önlenmiş olur.



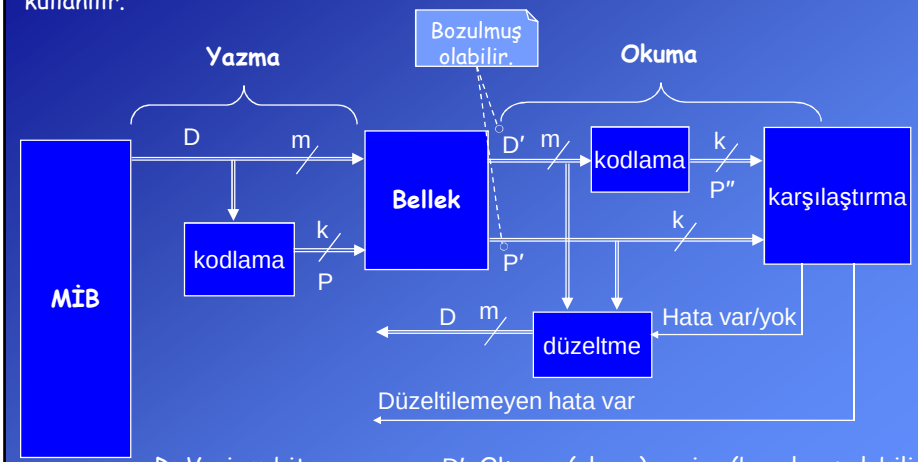
RAID 6

- İki eşlik bilgisi kullanılır böylece hata sezme miktarı arttırılmış olur.
- Fazladan bir diske gerek vardır.



Belleklerde Hata Sezme/Düzeltme

Güvenirlik gerekli sistemlerde hata düzeltici kodlama (*error correcting code* –ECC) kullanılır.



D: Veri, m bit
P: Eşlik, k bit
P": Okunan veriden hesaplanan eşlik

D': Okunan (alınan) veri (bozulmuş olabilir.)
P': Okunan (alınan) eşlik (bozulmuş olabilir.)

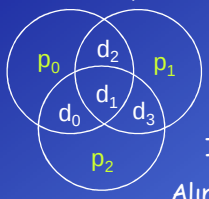
Bir bitlik hata düzelten Hamming Kodları (*Single Error Correction* – SEC)

Veri bitlerine, bir bitlik hatanın yerini bulmayı sağlayacak şekilde eşlik bitleri eklenir.

Eşlik bitlerini farklı şekillerde hesaplamak mümkündür.

Örnek: 4:7 Hamming kodu. (Richard Wesley Hamming (1915-1998), ABD)

4 bitlik veriye 3 bit eşlik eklenir toplam 7 bit iletilmiş (yazılmış) olur.



d_i: veri biti
p_i: eşlik biti

$p_0 = d_0 \oplus d_1 \oplus d_2$
$p_1 = d_1 \oplus d_2 \oplus d_3$
$p_2 = d_0 \oplus d_1 \oplus d_3$

İletilen kod sözcüğü: $d_0\, d_1\, d_2\, d_3\, p_0\, p_1\, p_2$ 4 bit veri + 3 bit eşlik

Alınan (okunan) sözcük: $d_0'\, d_1'\, d_2'\, d_3'\, p_0'\, p_1'\, p_2'$ bozulmuş olabilir

Alınan tarafta eşlikler yeniden hesaplanır:
$p_0'' = d_0' \oplus d_1' \oplus d_2'$
$p_1'' = d_1' \oplus d_2' \oplus d_3'$
$p_2'' = d_0' \oplus d_1' \oplus d_3'$

Alınan eşlikler ile hesaplanan eşlikler karşılaştırılır:
$s_0 = p_0' \oplus p_0''$
$s_1 = p_1' \oplus p_1''$
$s_2 = p_2' \oplus p_2''$

Sendrom bitlerinin (s_i) hepsi sıfırsa hata yok demektir. Sendrom sıfırdan farklı ise hatalı bitin yeri belirlenir ve tümlenerek düzeltilir.

Sendrom etkileşim tablosu:
Hangi sendrom bitinin hangi kod sözcüğü bitinden etkilendiğini gösterir.

	d_0	d_1	d_2	d_3	p_0	p_1	p_2
s_0	x	x	x		x		
s_1		x	x	x		x	
s_2	x	x		x			x

Sendrom Tablosu:

s_0 s_1 s_2	Anlamı
0 0 0	Hata yok
0 0 1	p_2 (bozulmuş)
0 1 0	p_1
0 1 1	d_3
1 0 0	p_0
1 0 1	d_0
1 1 0	d_2
1 1 1	d_1

Eşlik bitlerinin sayısının belirlenmesi:
Veri bitleri sayısı: m
Eşlik bitleri sayısı: k $2^k - 1$ bitlik bir kod sözcüğü içinde 1 bit hata düzeltilebilir.
Buna göre: $\mathbf{m + k \le 2^k - 1}$ olmalıdır.

Bir bitlik hata düzeltme, iki bitlik hataları sezme:
(*Single error correction – double error detection* SEC-DED)
Bir bitlik hataları düzeltmeye ek olarak iki bitlik hataları sezmek üzere kod sözcüğünün sonuna bir eşlik biti daha eklenir. Bu eşlik biti 1'lerin sayısını tek ya da çift yapacak şekilde seçilir.
Eğer sendrom sıfırdan farklı (hata var) ve ek eşlik biti "hata yok" sonucu veriyorsa çift sayıda hata olmuş demektir.
En yaygın kullanılan 64 + 7 + 1 bitlik kodlamadır. %12.5 fazlalık var.