Chapter 4

Bellek

Okuma Listesi

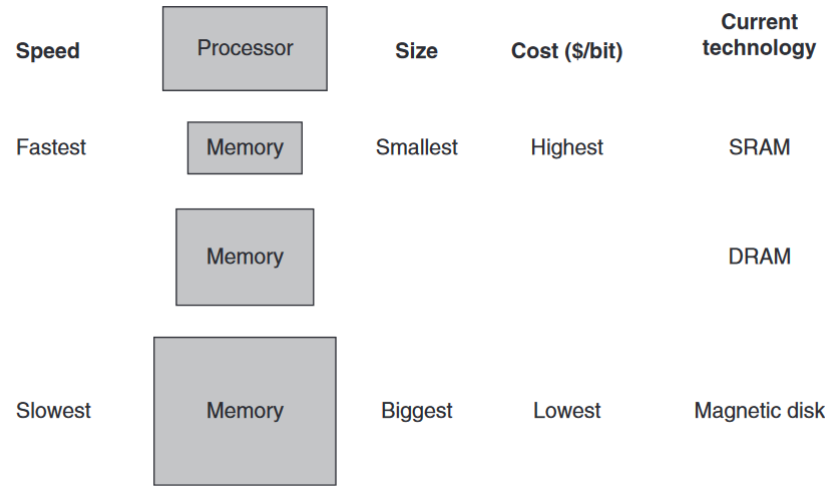
Gerekli

- Computer Organization and Design: The Hardware Software Interface [RISC-V Edition] David A. Patterson, John L. Hennessy
  - 5. Bölüm
- TOBB üniversitesi, Prof Dr. Oğuz ERGİN, Bilgisayar Mimarisi ve Organizasyonu dersi ders sunumları

Bellek Teknolojileri

Günümüz sistemlerinde 4 temel bellek teknolojisi kullanılır:

1. Durağan Rastgele Erişimli Bellek (SRAM)
2. Devingen Rastgele Erişimli Bellek (DRAM)
3. Flash
4. Manyetik Disk



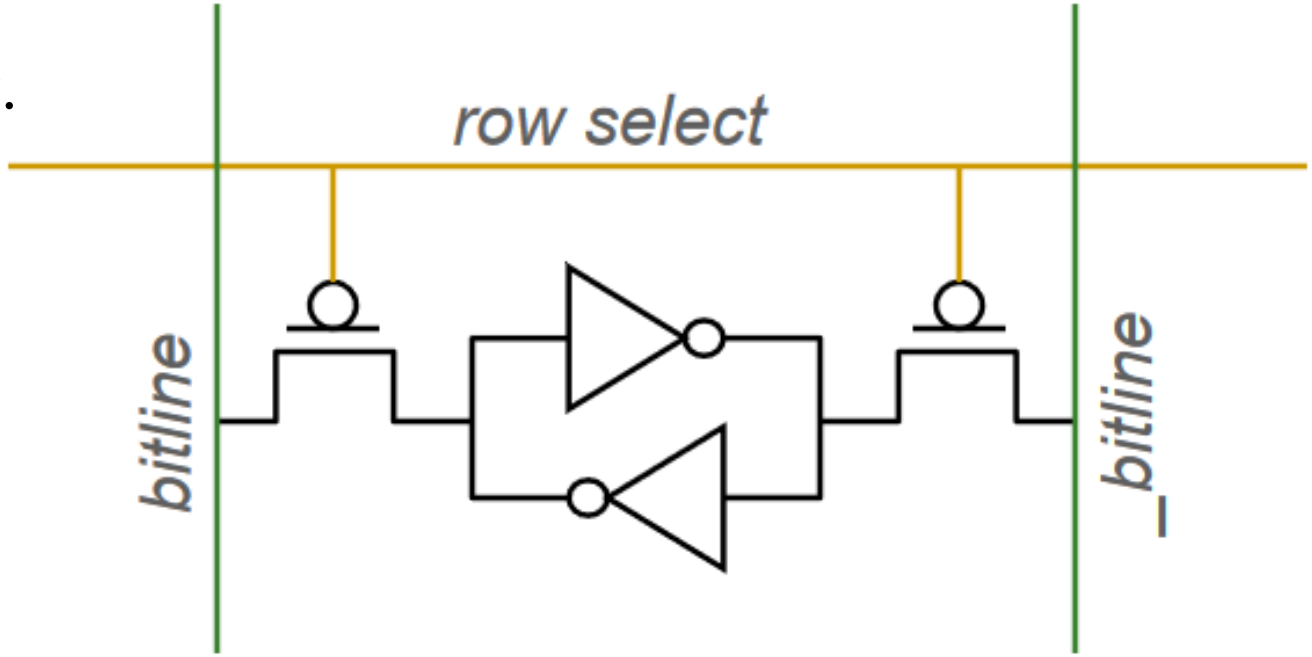
Ödünleşme

Teknoloji	Ortalama Erişim Zamanı	GiB Başına Fiyat (2012)
SRAM yarı iletken bellek	0.5-2.5 ns	$500 - $1000
DRAM yarı iletken bellek	50-70 ns	$10 - $20
Flash yarı iletken bellek	5.000 – 50.000 ns	$0.75 - $1.00
Manyetik Disk	5.000.000 – 20.000.000 ns	$0.05 - $0.10

1 GiB = 2^{10} MiB = 2^{20} KiB = 2^{30} Bayt

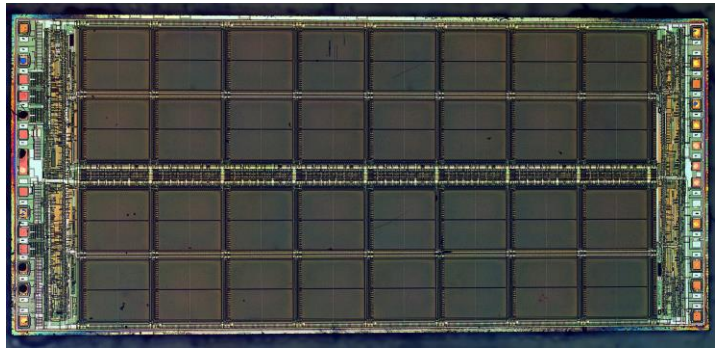
SRAM

- 2 tersleyici bir bit tutar:
  - 4 transistör saklama için kullanılır
  - 2 transistör erişim için kullanılır.
- Sabit erişim süresine sahiptir.
- Önbelleklerde kullanılır.



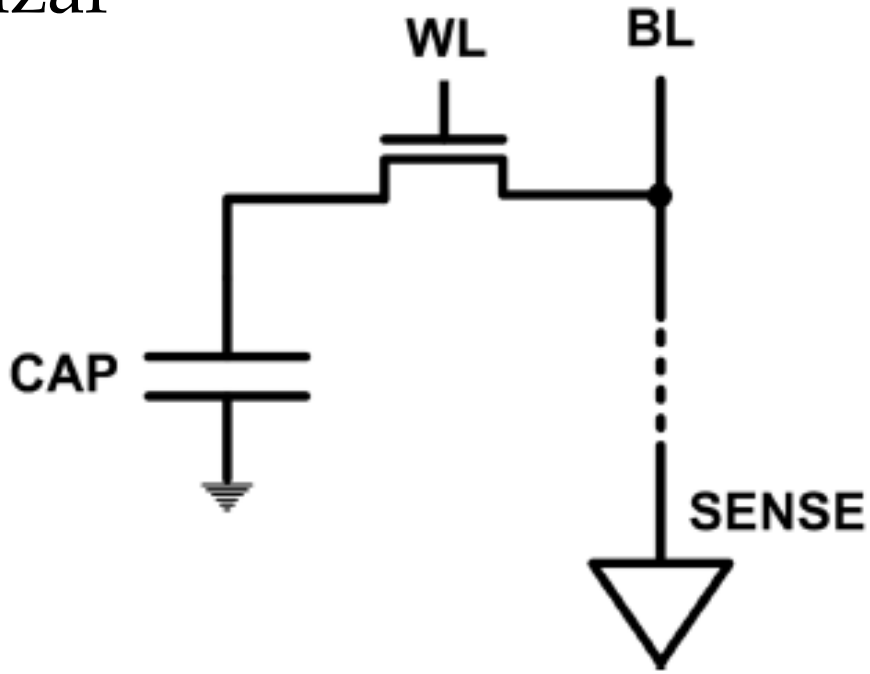
https://commons.wikimedia.org/wiki/File:Hynix_HY62V8100B_SRAM-7313.jpg
https://users.ece.cmu.edu/~omutlu/pub/onur-ACACES2013-Lecture1-dram-basics-and-scaling-afterlecture.pdf

DRAM



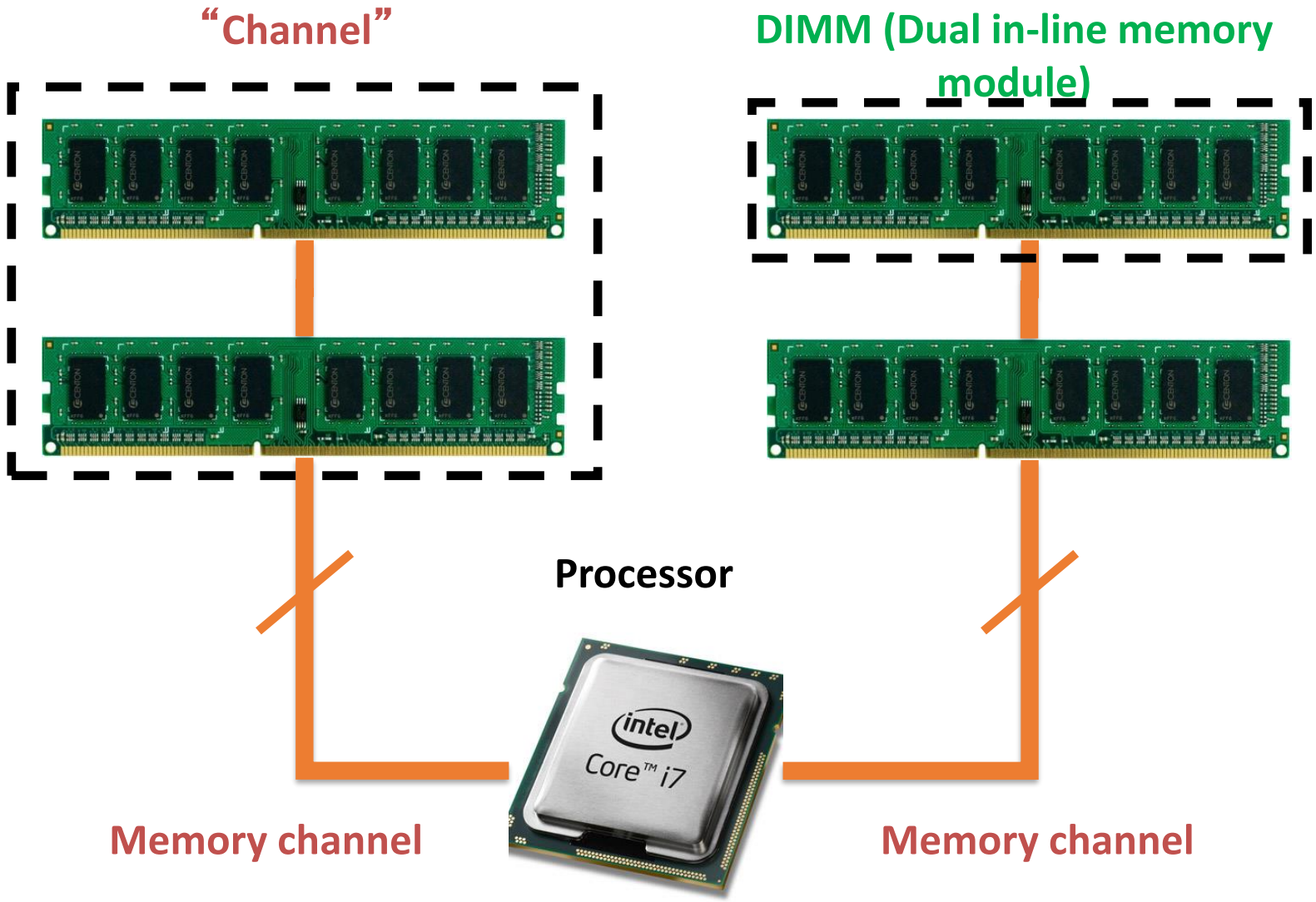
- Saklama için kapasitör kullanır.
- Kapasitör içerisindeki yük zamanla dışarı sızar bu nedenle belirli periyotlarla **tazelenmesi** gerekir.

Yıl	Yonga Büyüklüğü	GiB başına Fiyat	Erişim Süresi
2000	246 Mebibit	$1.000	55 ns
2004	512 Mebibit	$250	50 ns
2007	1 Gibibit	$50	45 ns
2010	2 Gibibit	$30	40 ns
2012	4 Gibibit	$1	35 ns

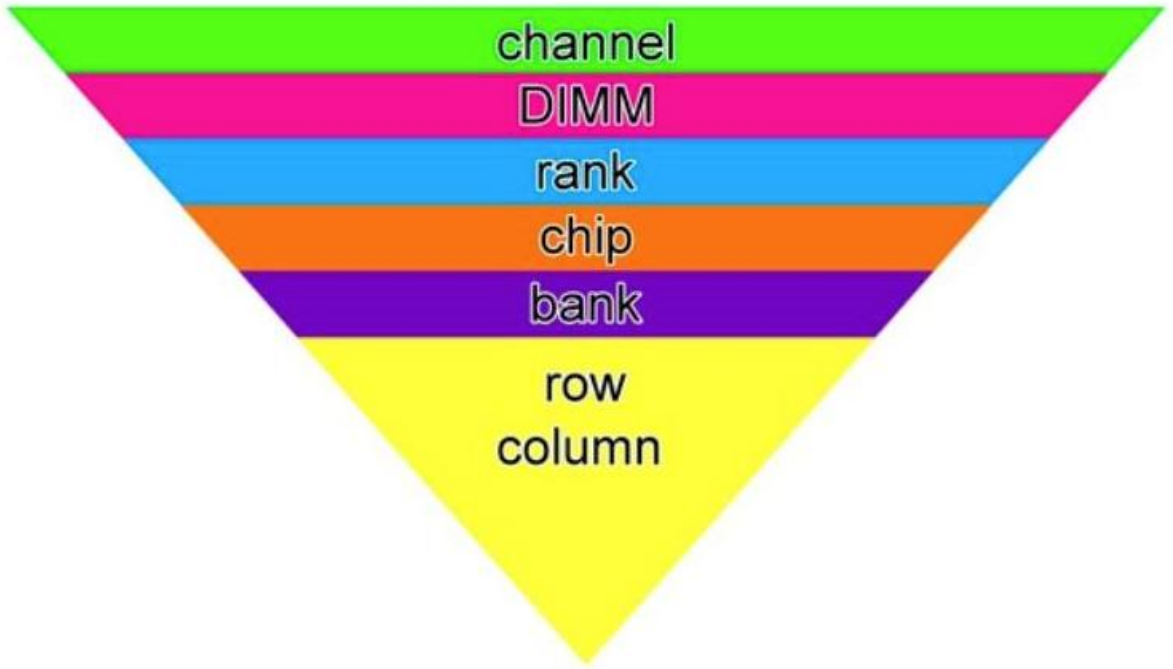


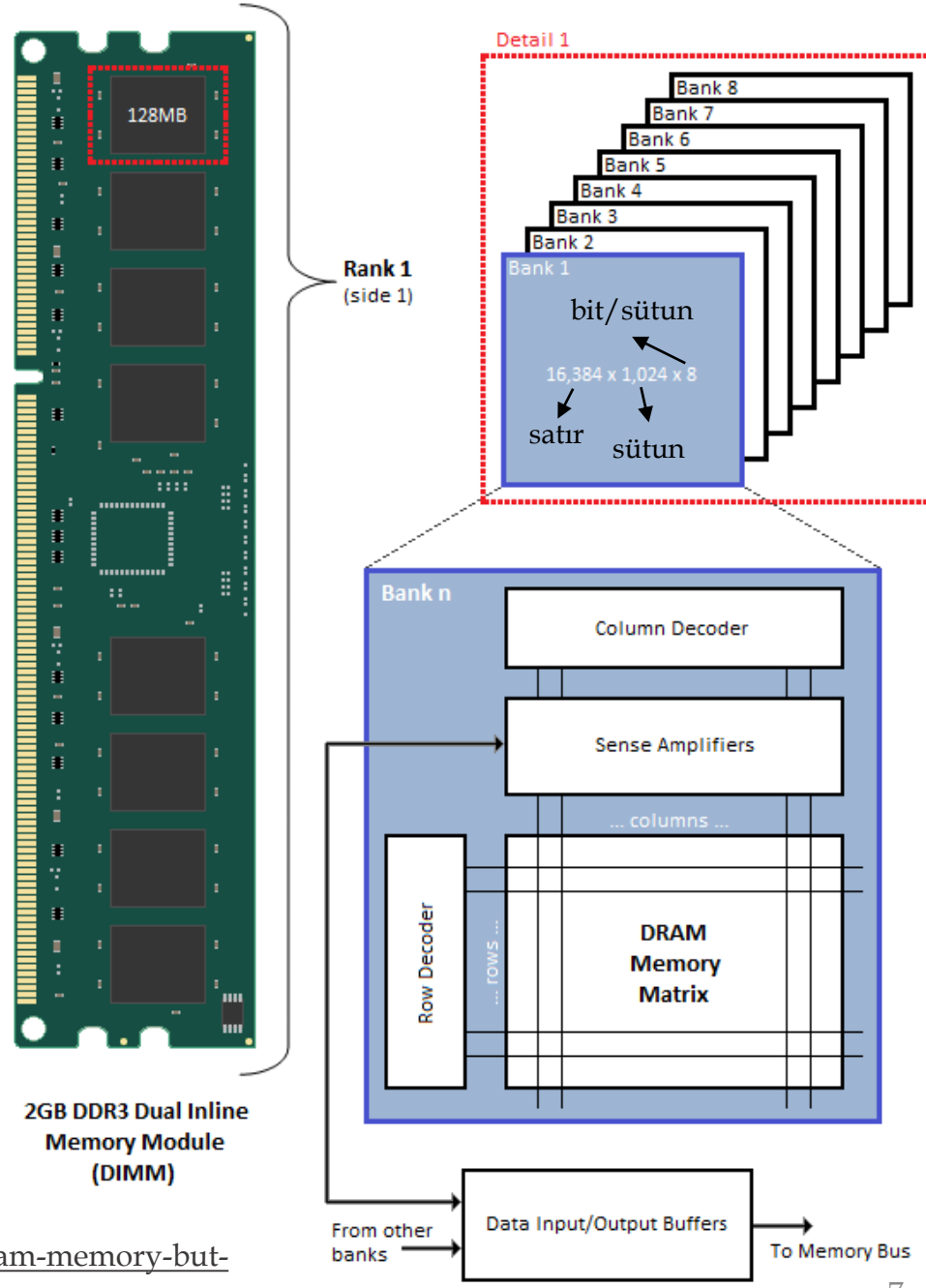
https://en.wikipedia.org/wiki/Dynamic_random-access_memory#/media/File:MT4C1024-HD.jpg
https://people.inf.ethz.ch/omutlu/pub/onur-ACACES2018-Lecture1-Topic1-MemoryTrendsAndBasics-July-9-2018-afterlecture-with-backup.pdf

DRAM Organizasyonu



DRAM Organizasyonu





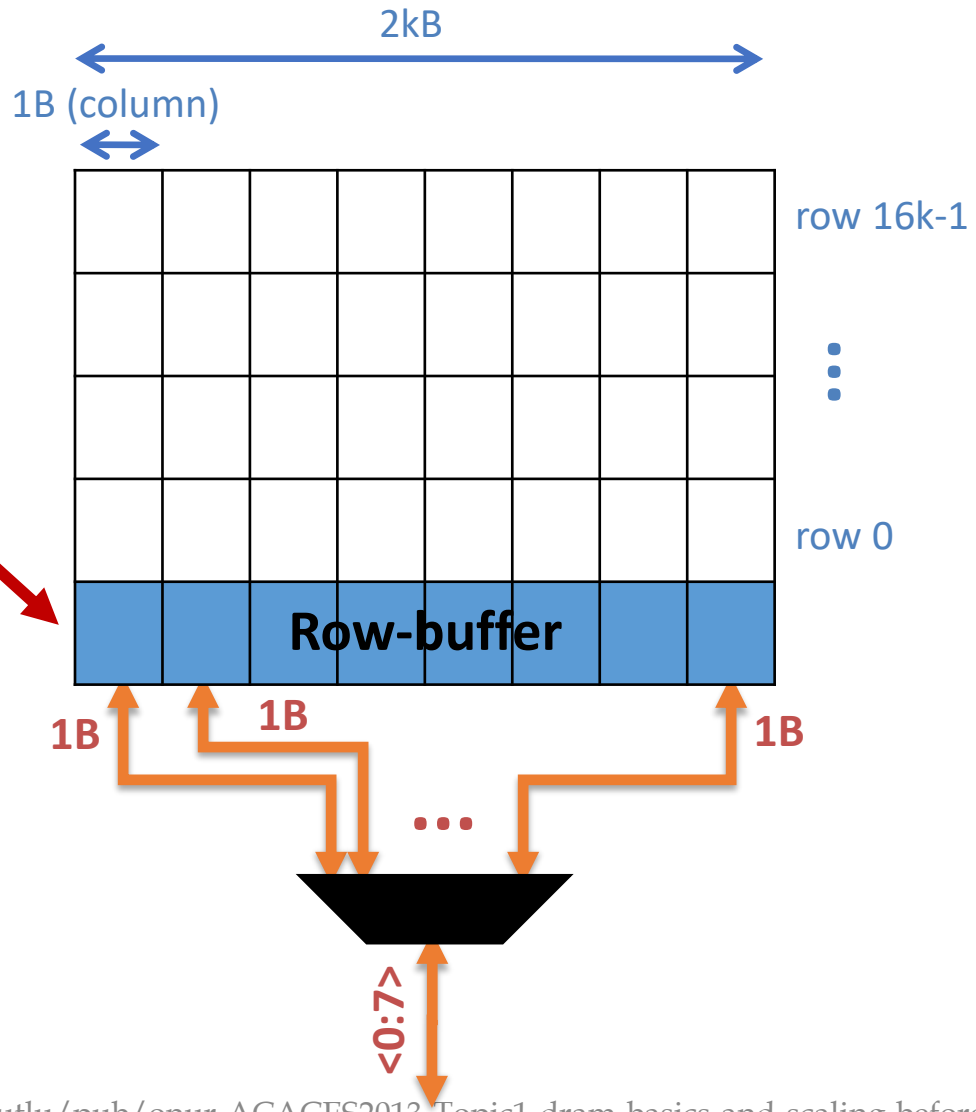
https://www.anandtech.com/show/3851/everything-you-always-wanted-to-know-about-sdram-memory-but-were-afraid-to-ask/2

https://programmersought.com/article/4267347843/

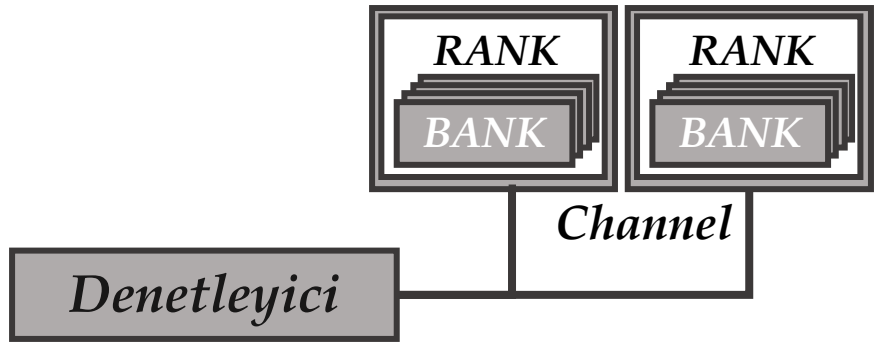
DRAM Organizasyonu

DRAM aynı satıra gelen istekleri ara bellekten karşılar.

Erişim zamanını kısaltır.

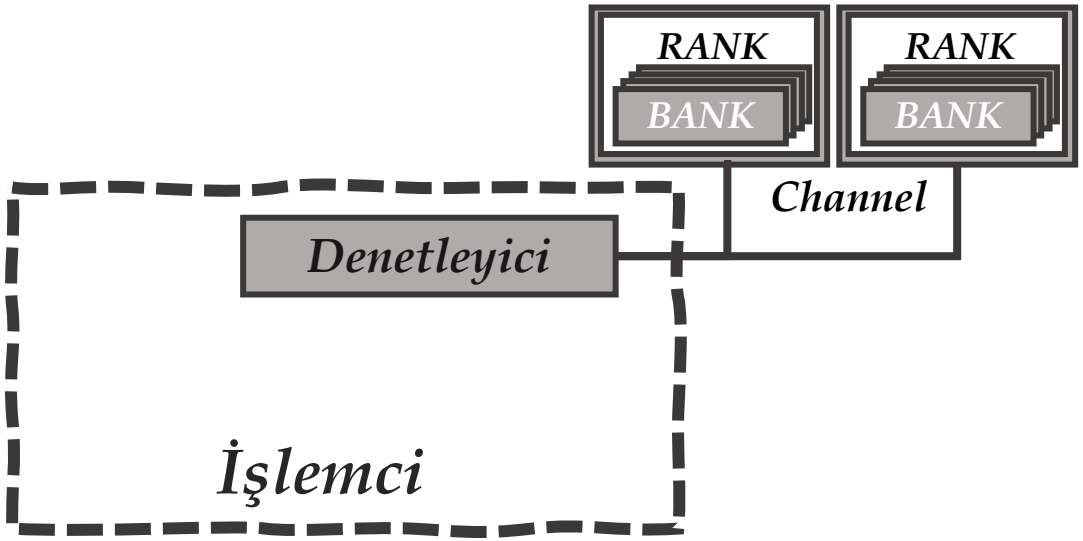


DRAM Denetleyicisi

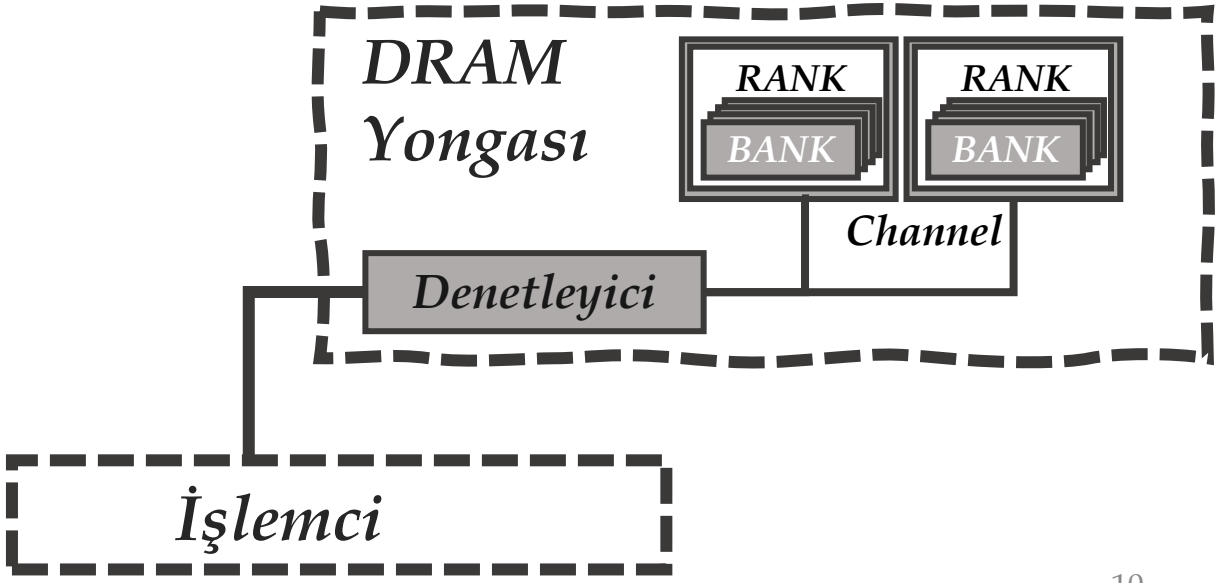


- Yapılan işlemlerin doğruluğunu sağlar.
  - Tazeleme ve zamanlamadan sorumludur.
- Denetleyiciye gelen buyrukları DRAM komutlarına dönüştürür.
- İki şekilde yerleştirilebilir:

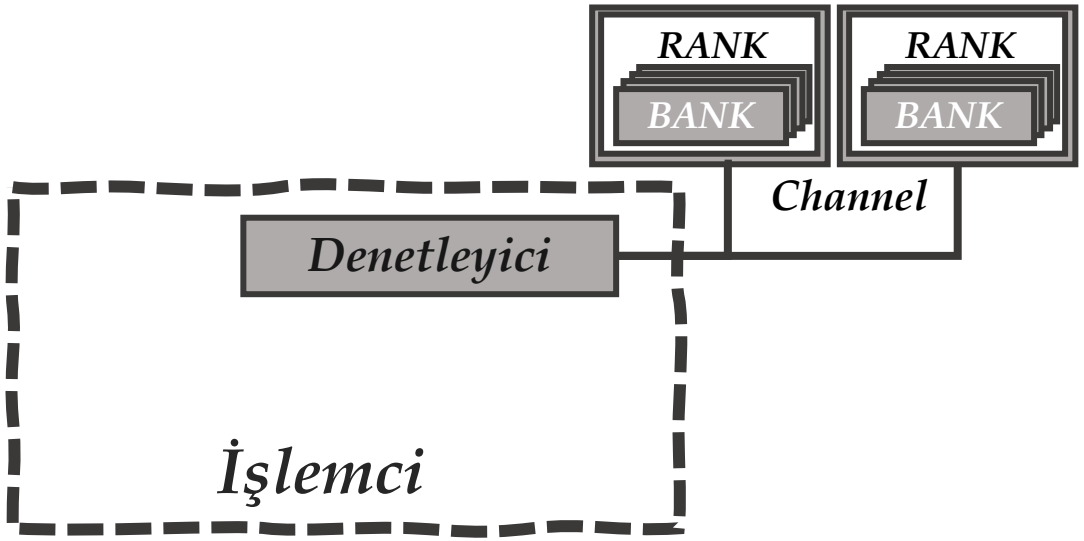
(1) İşlemci içerisinde

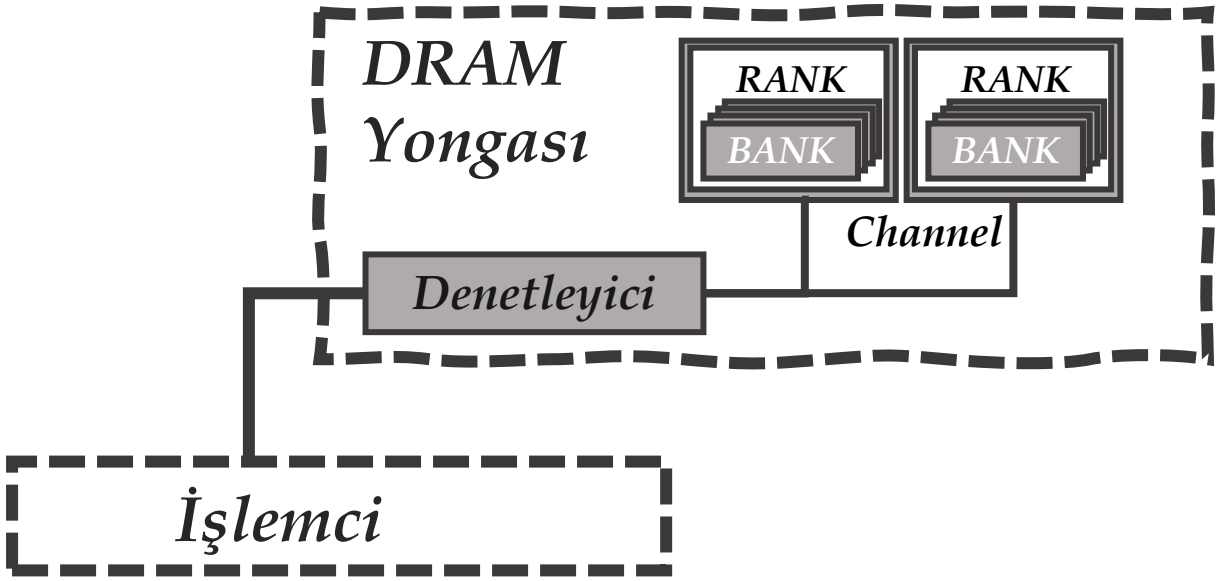


(2) DRAM yongası üzerinde



DRAM Denetleyicisi





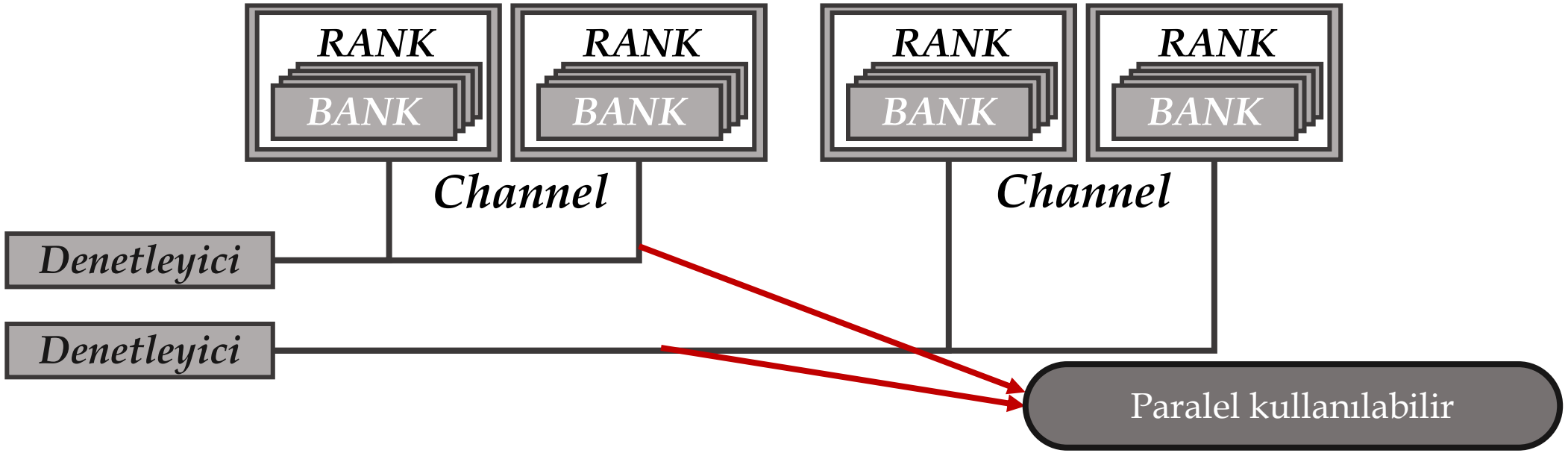
- **Bellek erişim gecikmesi daha az**
- Çekirdekler ve denetleyici arasındaki **bant genişliği daha yüksek**

- Farklı DRAM aygıtları takılması **daha kolay** (esneklik)
- İşlemci yongasında **daha az enerji tüketimi** olur

Biniştirme

Birden fazla kanal olması paralel DRAM erişimine olanak sağlar.

- Aynı şekilde birden fazla bank olması da paralelliği artırır. Ancak farklı kanallar farklı veri yollarına sahip olduğu için bant genişliği açısından daha avantajlıdır.



Paralellik oluşturmak için bir programa ait veriler farklı şekillerde DRAM'e yazılabilir.

Flash Bellek

Flash bellekler EEPROM (*electrically erasable programmable read-oly memory*) tipinde belleklerdir.

Yazılan verileri **kalıcı olarak** saklayabilirler.

Diğer teknolojilerden farklı olarak **yazma işlemleri** bellekteki bitlerin **saklama özelliğini kaybetmesine** yol açabilir.

- Belleğin tamamında veri kaybının aynı zamanlarda yaşanması için denetleyici tarafından özel veri dağıtma teknikleri kullanılır.

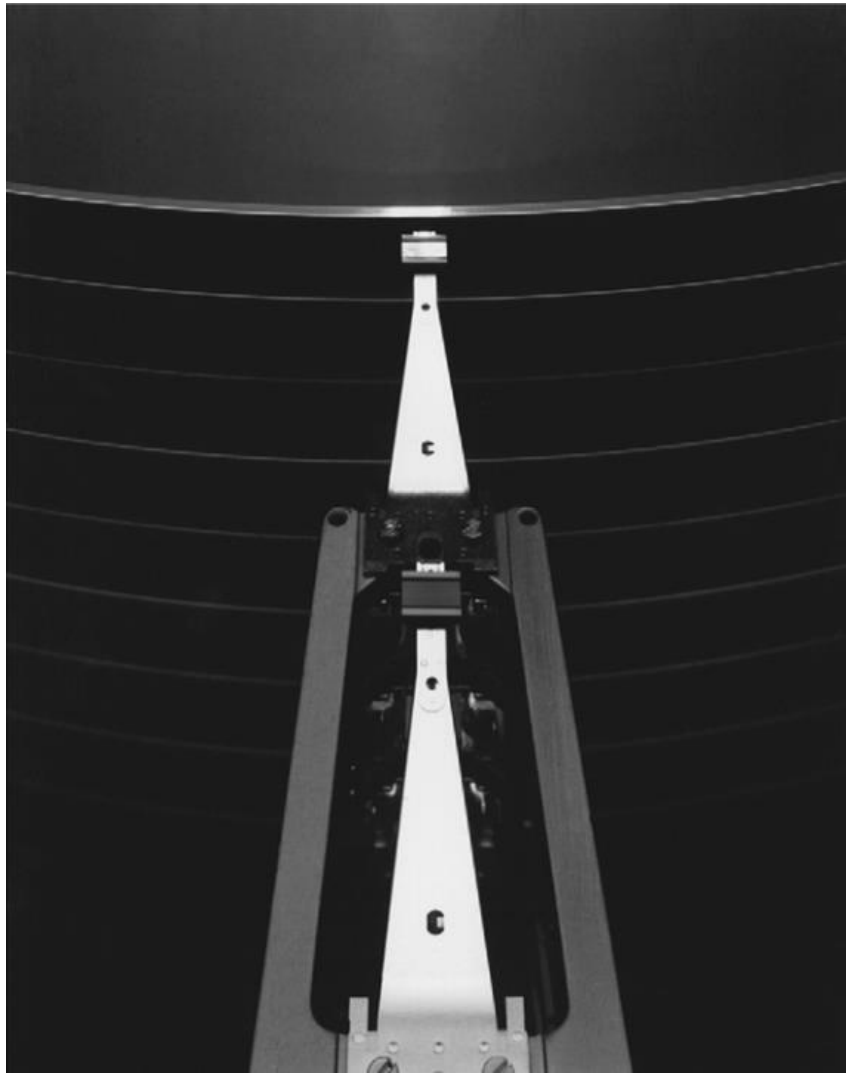
Manyetik Teker (Disk)

Kalıcı bellek olarak kullanılır.

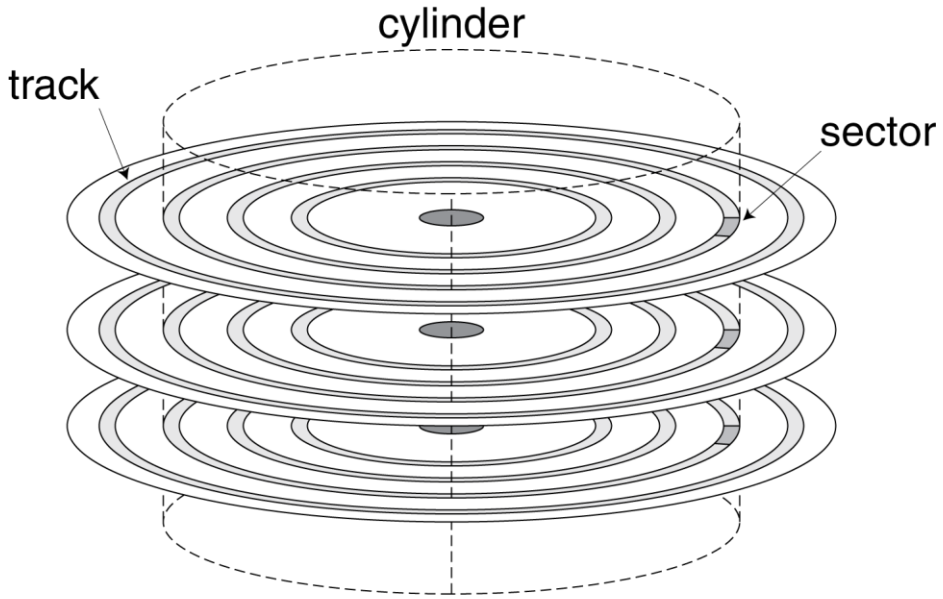
Mıknatıslama yöntemi ile veri disklerin üzerine yazılır.

Disklerin bir dönüş hızı vardır ve yazma ile okuma işlemleri hareket edilebilir bir kol aracılığı ile yapılır.

Bir manyetik diskin hızında fiziksel dönüş hızı, erişim süresi, ve aktarma hızı olmak üzere üç özellik etkileşimli olarak rol oynar



Manyetik Teker (Disk)



Yerellik

Belleğe erişim her zaman tamamen rastgele olmaz, genellikle bir program aynı süre zarfında verilerinden bir kısmına erişir.

Programların bellek erişimleri **yerellik prensibi** (-ing. principle of locality) ile açıklanır.

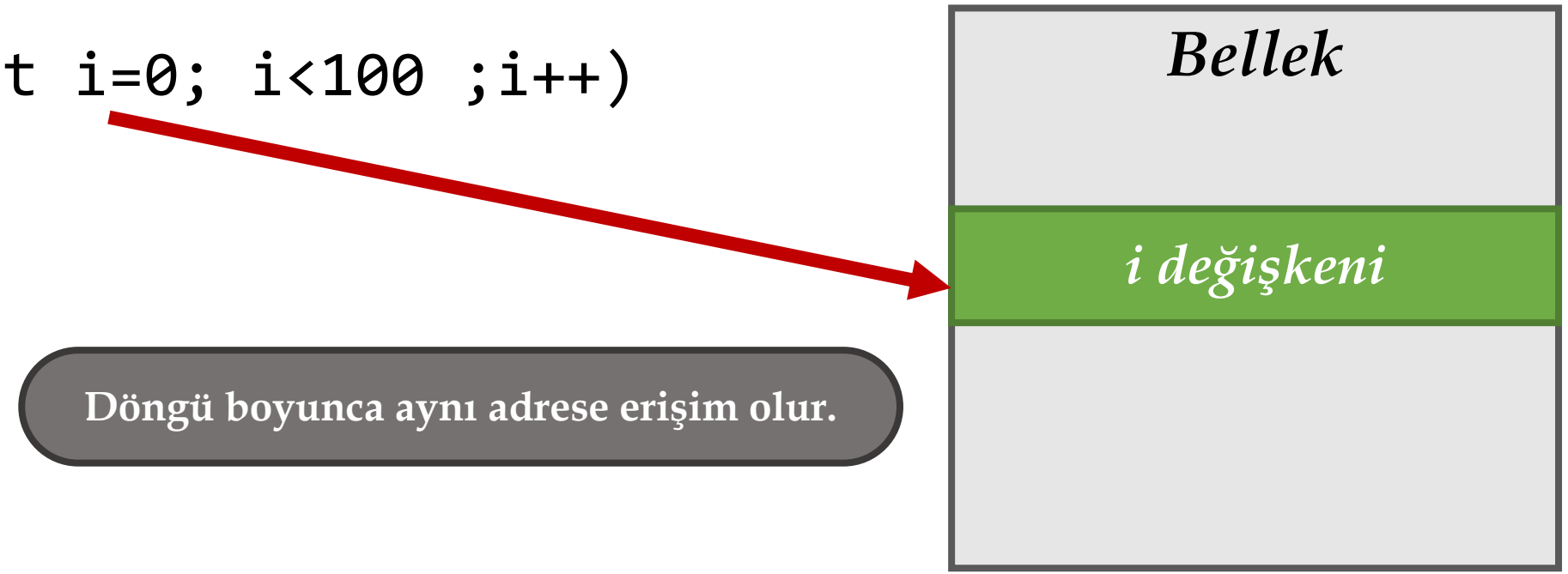
İki tip yerellik vardır:

1. Zamanda Yerellik
2. Alanda Yerellik

Zamanda Yerellik

Aynı veriye belirli bir süre zarfında erişilmesine **zamanda yerellik** (-ing. temporal locality) denir.

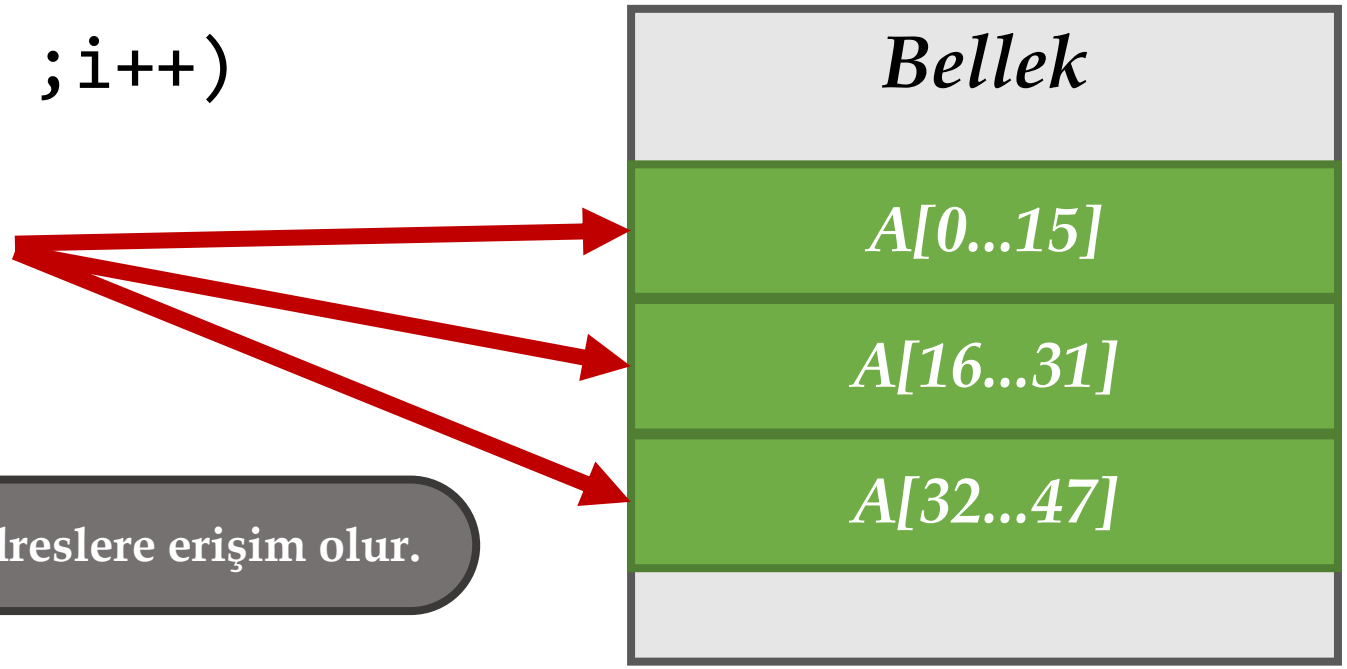
```
for (int i=0; i<100 ;i++)
{
…
}
```



Alanda Yerellik

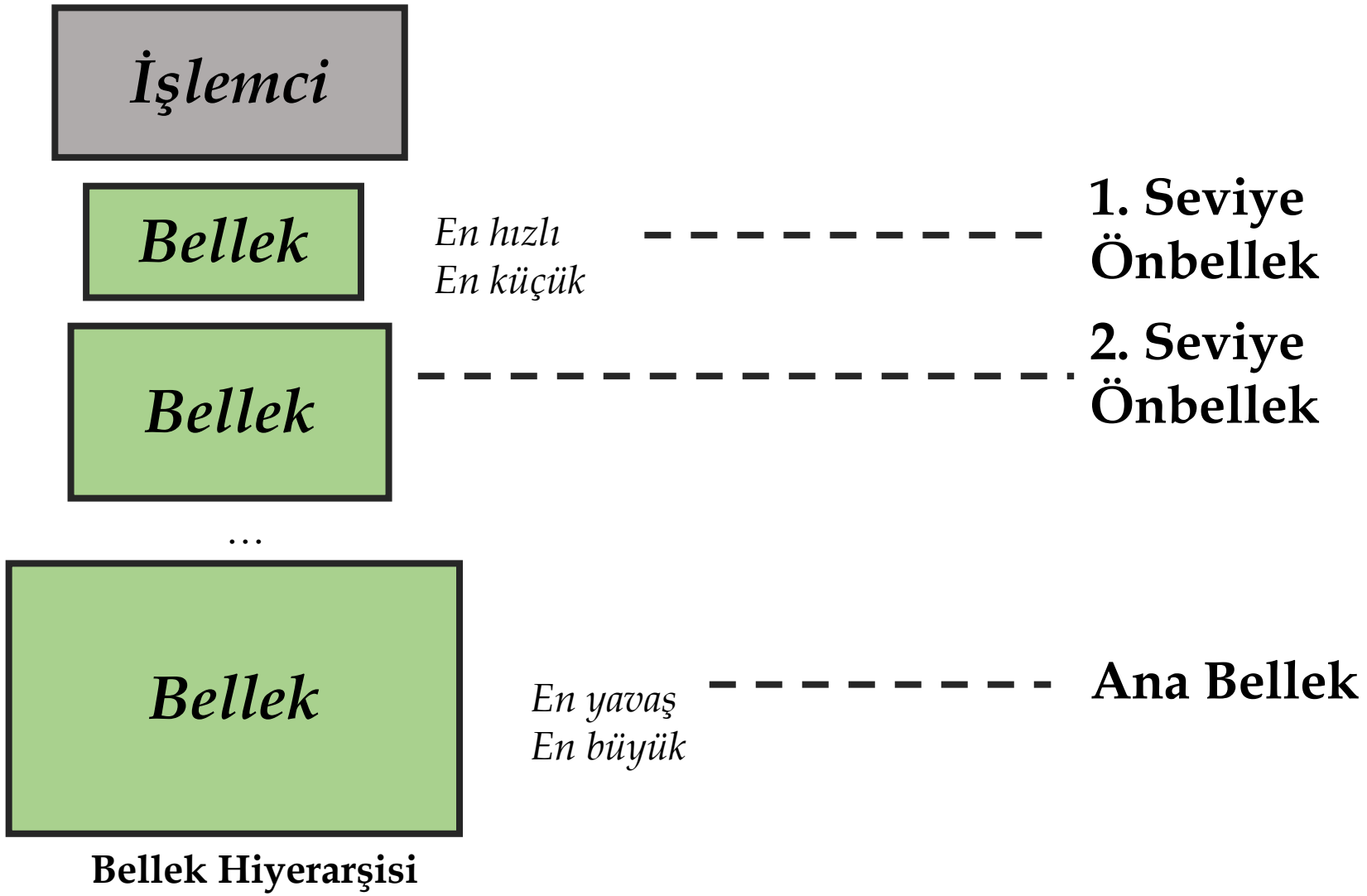
Birbirine yakın bellek adreslerine yazılmış verilere erişilmesine **alanda yerellik** (-ing. spatial locality) denir.

```
for (int i=0; i<48 ;i++)
{
      int j = A[i];
      ...
}
```



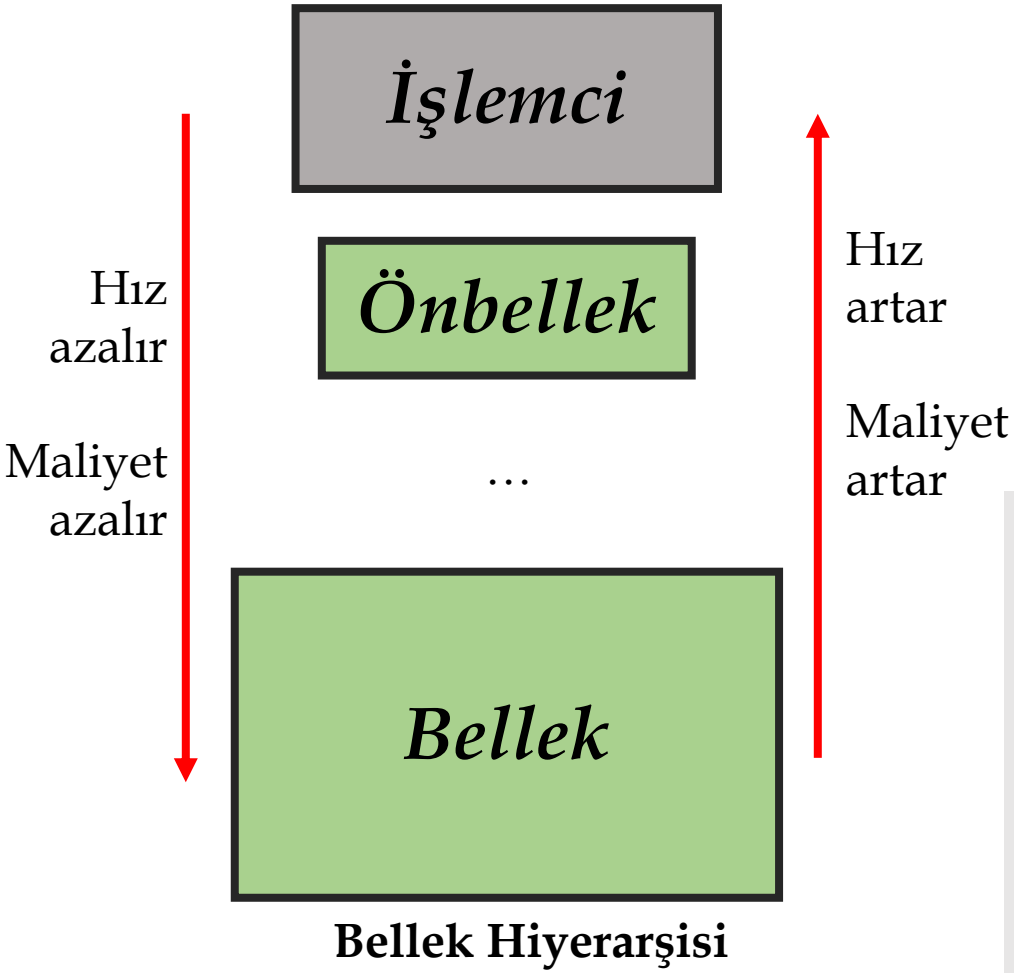
Döngü boyunca yakın adreslere erişim olur.

Bellek Hiyerarşisi



Bellek Hiyerarşisi

Önbellek



Bellek Hiyerarşisi

Önbellek işlemcinin en kısa zamanda erişmesi gereken verileri tutar.

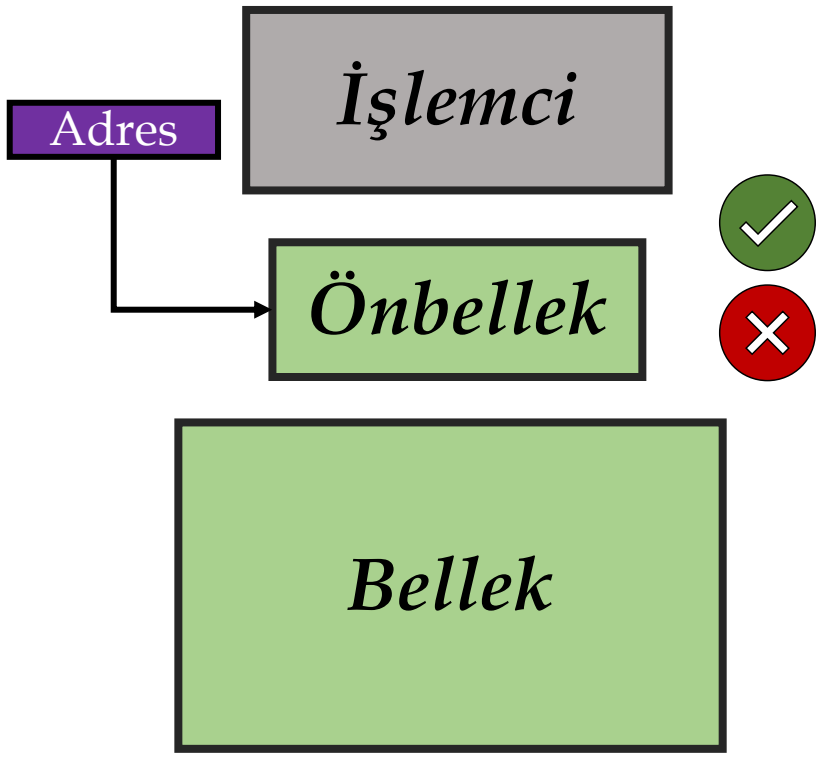
Önbellek erişilen veride yerellikten yararlanır.

Hatırlatma:

Aynı veriye belirli bir süre zarfında erişilmesine **zamanda yerellik** (-ing. temporal locality) denir.

Birbirine yakın bellek adreslerine yazılmış verilere erişilmesine **alanda yerellik** (-ing. spatial locality) denir.

Önbellek Temel Kavramlar



Bellek Hiyerarşisi

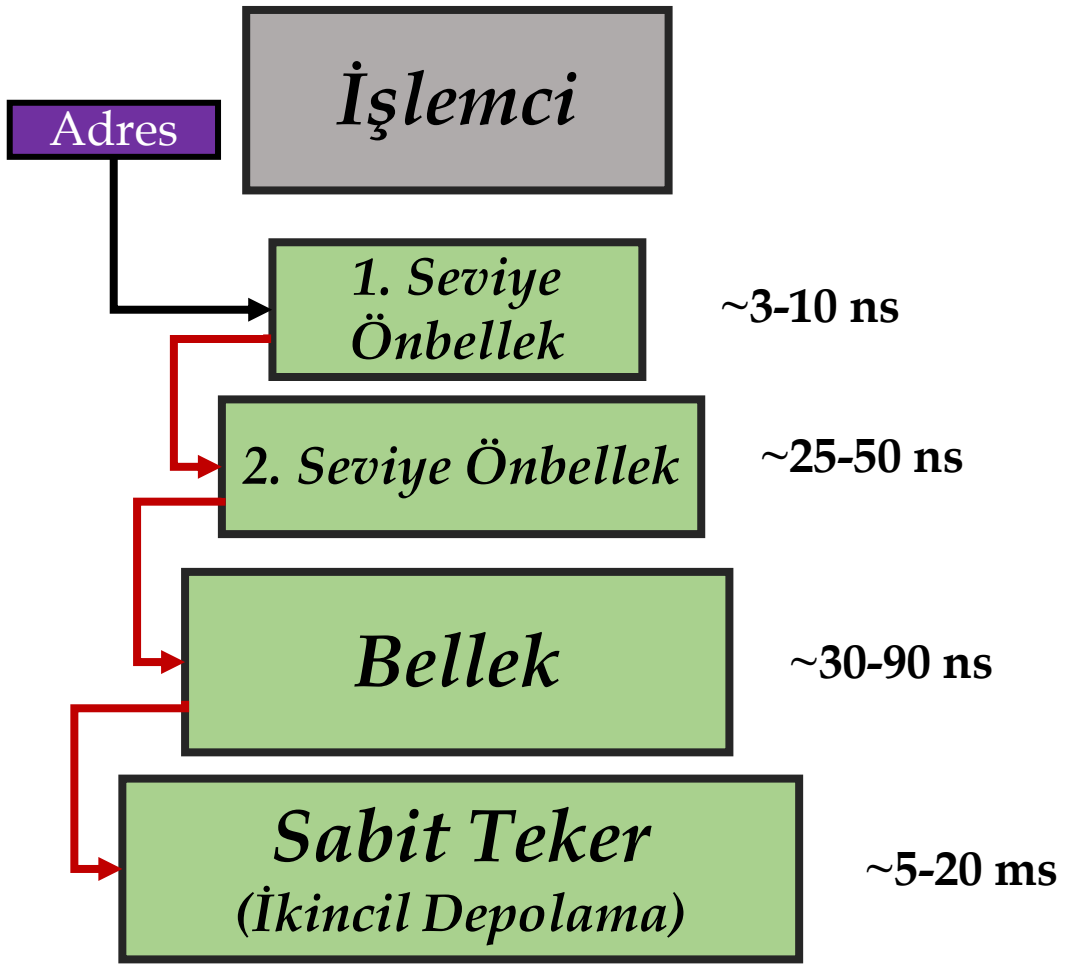
Önbellek ana bellekteki verilerin bir kısmına sahiptir. Veriyi almak için bir adresle önbelleğe gidildiğinde iki durum oluşur:

1. Verinin bulunması (-ing. hit)
2. Verinin bulunamaması (-ing. miss)

$$\text{Bulma Oranı} = \frac{\text{Bulma sayısı}}{\text{Toplam erişim sayısı}}$$

$$\text{Bulamama Oranı} = \frac{\text{Bulamama sayısı}}{\text{Toplam erişim sayısı}}$$

Önbellek Temel Kavramlar



Bellekte istenilen verinin bulunduğuna dair bilgi gelene kadar geçen süreye **«bulma zamanı»** denir.

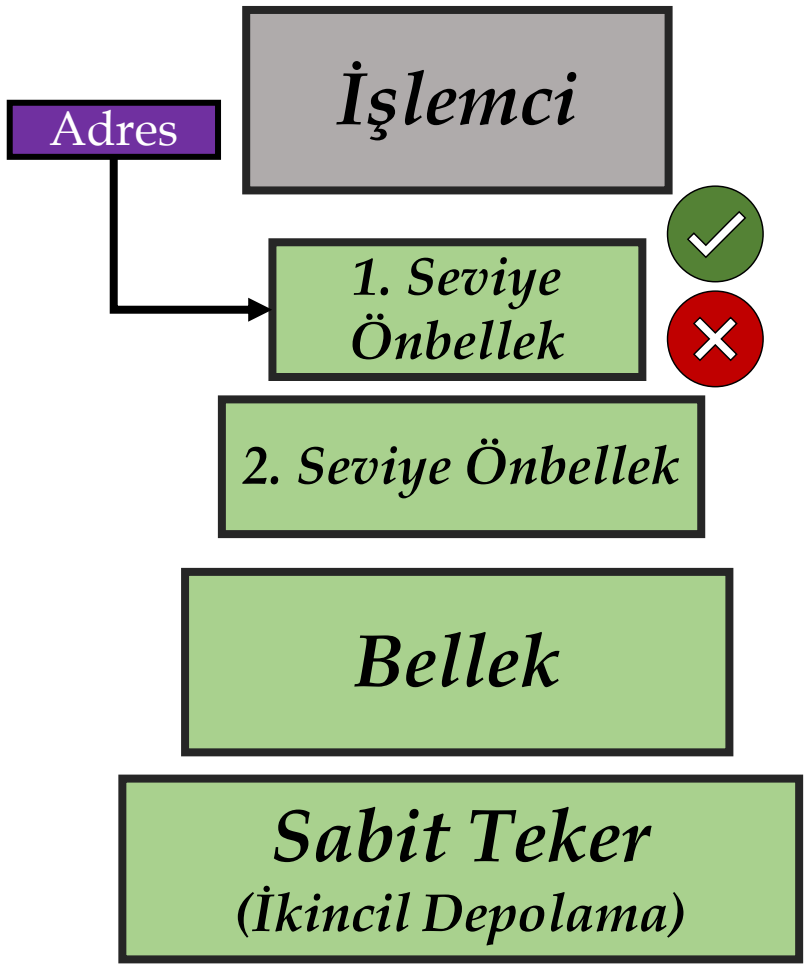
Bulma Zamanı = Belleğe Erişim Süresi + Verinin bulunup bulunmadığının belirlenme süresi

Bellekte istenilen veri bulunamazsa bir sonraki aşamadaki belleğe gidilir.

Verinin getirilmesi sırasında geçen süreye **«bulamama gecikmesi»** denir.

Bulamama Gecikmesi = (Yer yoksa) Önbellekte Yer Açılma Süresi + Verinin getirilmesi süresi

Önbellek Temel Kavramlar



Bellekte bir verinin bulunup bulunmadığını nasıl anlarız?

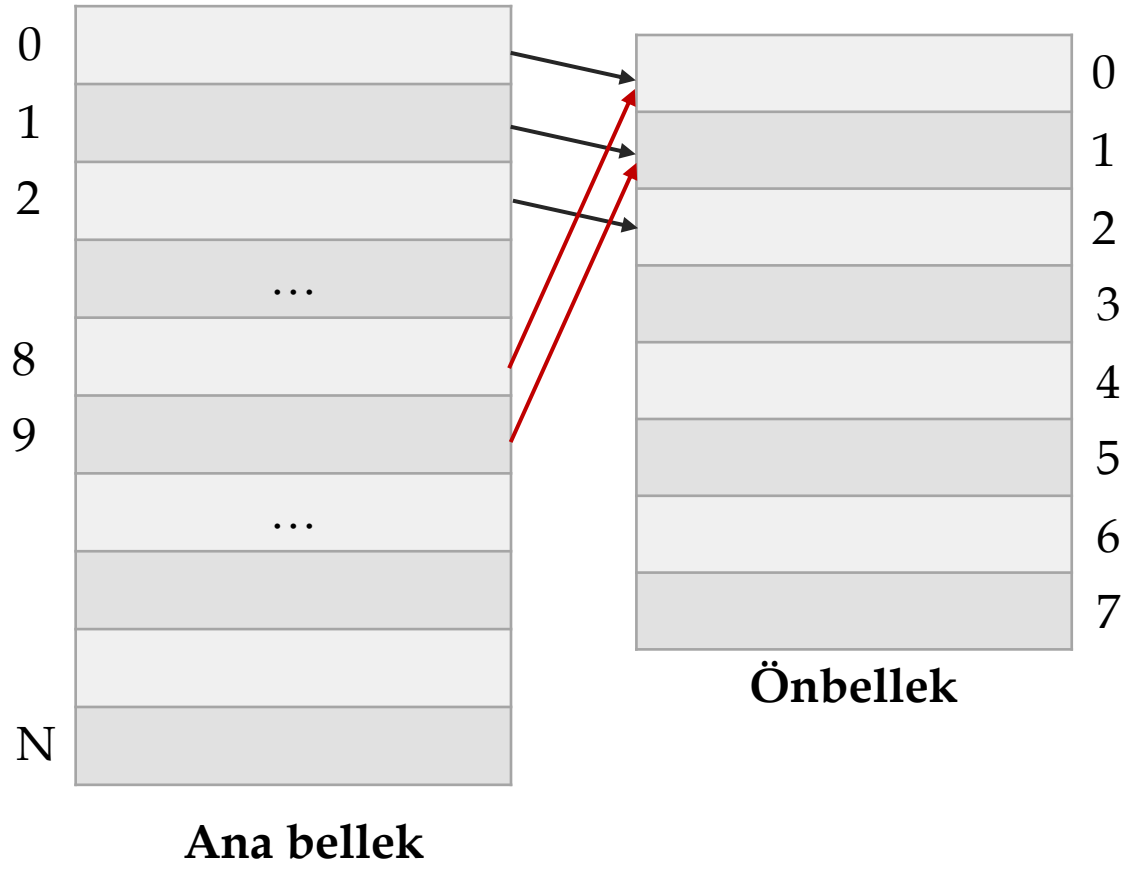
Bellekte bulunan bir verinin nerede olduğunu nasıl buluruz?

Bu soruların cevabı önbellek tasarımı ile ilgilidir.

Doğrudan Eşlemeli Önbellek

Doğrudan eşlemeli önbellek (-ing. direct mapped cache) verileri adreslerine göre yerleştirir.

Bir veri bellekte yalnızca bir yerde olabilir.



Önbellek ana bellekteki verinin yalnızca bir kısmını tutabilecek büyüklüktedir.
Veri yerleştirme yöntemi:

Adres = [Veri adresi] mod [Önbellek boyutu]

Önbellek boyutu 2'nin kuvveti ise en anlamsız ***log_2(önbellek boyutu)*** biti önbellekte adresleme için kullanabiliriz.

Önbellek Boyutu = 8, $\log_2 8 = 3$
$Adres_1 = 1 = 00\mathbf{001}$
$Adres_2 = 9 = 01\mathbf{001}$
} İkisi de 001 yani 1'e yazılır.

Doğrudan Eşlemeli Önbellek

Önbellek Boyutu = 8, $\log_2 8 = 3$

- $\text{Adres}_1 = 1 = 00\mathbf{001}$
- $\text{Adres}_2 = 9 = 01\mathbf{001}$

} İkisi de aynı yere yazılır.

Önbellekte hangi verinin bulunduğunu nasıl anlayabiliriz?

Önbellekte hangi verinin bulunduğunu anlamak için **etiket** (-ing. tag) alanı kullanılır.

$\text{Adres}_1 = 1 = \mathbf{00}001$
$\text{Adres}_2 = 9 = \mathbf{01}001$

Doğrudan Eşlemeli Önbellek

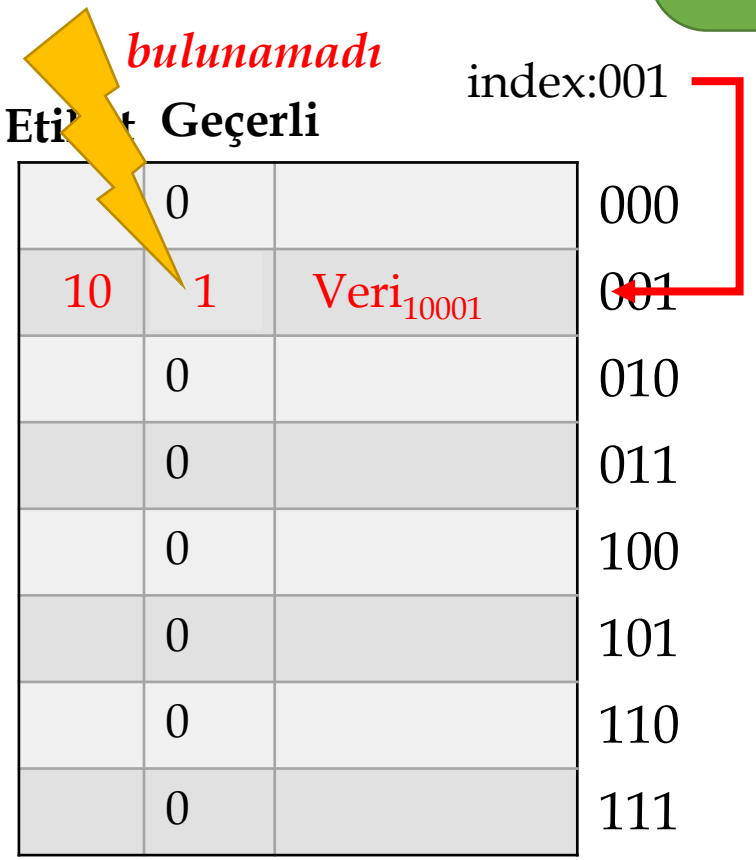
Önbellekte tutulması gereken bir diğer bilgi ise verinin geçerli olup olmadığı bilgisidir.

Örneğin bilgisayar ilk açıldığında önbellekte bulunan veri geçerli değildir, anlamsızdır. Önbellekteki veri yerine ana bellekteki veri kullanılmalıdır.

Bunun için her satır için **geçerli** (-ing. valid) biti tutulur.

Örnek

Bir sonraki seviyeden veri getirilir



Sırası ile aşağıdaki adreslere erişiliyor.

Önbelleğin son durumu ne olur?

- $(10001)_2$
- $(11000)_2$
- $(00100)_2$
- $(11101)_2$
- $(11001)_2$

etiket ← $(10001)_2$ → index

Örnek

bulunamadı

Bir sonraki seviyeden veri getirilir

Etiket	Geçerli		
11	1	$Veri_{11000}$	000
10	1	$Veri_{10001}$	001
	0		010
	0		011
	0		100
	0		101
	0		110
	0		111

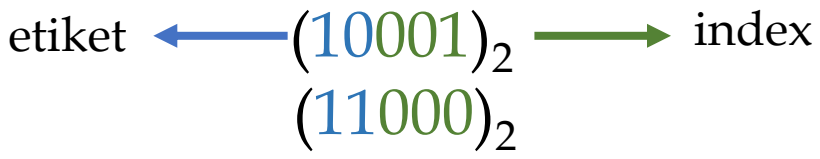
index:000

Önbellek

Sırası ile aşağıdaki adreslere erişiliyor.

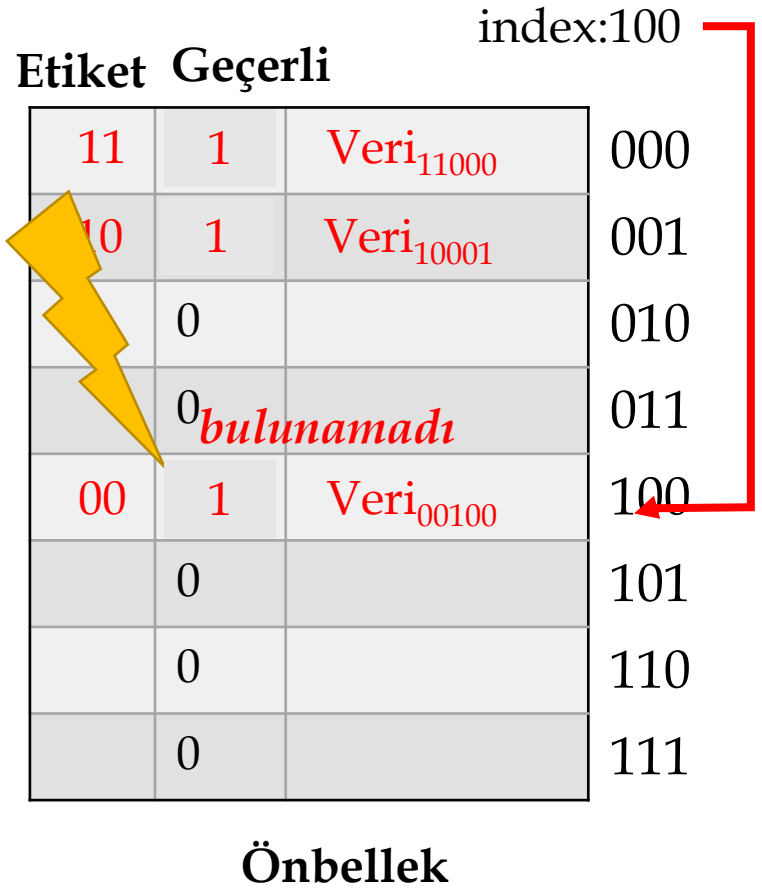
Önbelleğin son durumu ne olur?

- $(10001)_2$
- $(11000)_2$
- $(00100)_2$
- $(11101)_2$
- $(11001)_2$



Örnek

Bir sonraki seviyeden veri getirilir



Sırası ile aşağıdaki adreslere erişiliyor.

Önbelleğin son durumu ne olur?

- $(10001)_2$
- $(11000)_2$
- $(00100)_2$
- $(11101)_2$
- $(11001)_2$

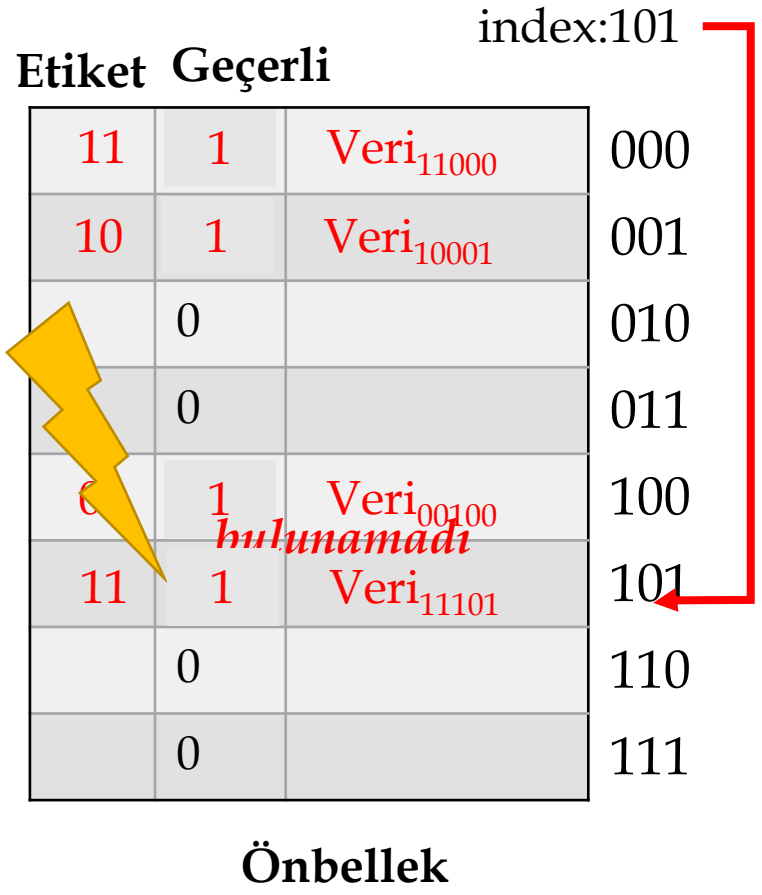
etiket ← $(10001)_2$ → index

$(11000)_2$

$(00100)_2$

Örnek

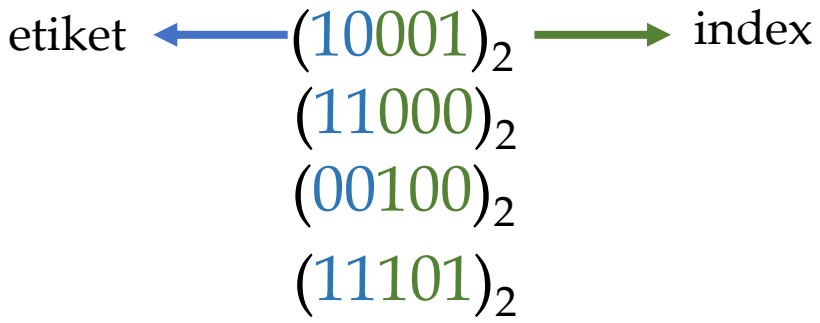
Bir sonraki seviyeden veri getirilir



Sırası ile aşağıdaki adreslere erişiliyor.

Önbelleğin son durumu ne olur?

- $(10001)_2$
- $(11000)_2$
- $(00100)_2$
- $(11101)_2$
- $(11001)_2$



Örnek

Var olan veri bir sonaki seviye belleğe yazılır. Bir sonraki seviyeden istenen veri getirilir

Etiketler aynı değil: bulunamadı

index:001

Etiket	Geçerli		
11	1	$Veri_{11000}$	000
11	1	$Veri_{11001}$	001
	0		010
	0		011
00	1	$Veri_{00100}$	100
11	1	$Veri_{11101}$	101
	0		110
	0		111

Önbellek

Sırası ile aşağıdaki adreslere erişiliyor.

Önbelleğin son durumu ne olur?

- $(10001)_2$
- $(11000)_2$
- $(00100)_2$
- $(11101)_2$
- $(11001)_2$

etiket ← $(10001)_2$ → index

$(11000)_2$

$(00100)_2$

$(11101)_2$

$(11001)_2$