

# BM-311 Bilgisayar Mimarisi

---

Hazırlayan: M.Ali Akcayol  
Gazi Üniversitesi  
Bilgisayar Mühendisliği Bölümü

## Konular

---

- **Hafıza sistemleri karakteristikleri**
- Hafıza hiyerarşisi
- Önbellek prensipleri
- Önbellek tasarım bileşenleri
  - Cache size
  - Mapping function
  - Replacement algorithms
  - Write policy
  - Line size
  - Number of caches

## Hafıza sistemleri karakteristikleri

<b>Location</b> Internal (e.g., processor registers, cache, main memory) External (e.g., optical disks, magnetic disks, tapes)	<b>Performance</b> Access time Cycle time Transfer rate
<b>Capacity</b> Number of words Number of bytes	<b>Physical Type</b> Semiconductor Magnetic Optical Magneto-optical
<b>Unit of Transfer</b> Word Block	<b>Physical Characteristics</b> Volatile/nonvolatile Erasable/nonerasable
<b>Access Method</b> Sequential Direct Random Associative	<b>Organization</b> Memory modules

## Hafıza sistemleri karakteristikleri

### Location

- **Internal:**
  - Register'lar, control unit memory, cache, main memory.
- **External:**
  - HDD, tape, I/O kontrol ile erişilenler.

## Hafıza sistemleri karakteristikleri

### Capacity

- **Number of words:**
  - Toplam satır sayısını ifade eder.
  - Genellikle internal memory için kullanılır.
  - Bir word 8, 16 veya 32 bit olabilir.
- **Number of bytes:**
  - External depolama birimlerinde kapasite byte olarak ifade edilir.

## Hafıza sistemleri karakteristikleri

### Unit of transfer

- **Word:**
  - Bir seferde bir word boyutunda veri okunur veya yazılır.
  - Main memory için kullanılır.
- **Block:**
  - Bir seferde bir blok veri okunur veya yazılır.
  - External depolama birimleri için kullanılır.

## Hafıza sistemleri karakteristikleri

### Access method

- **Sequential:** Okuma/yazma mekanizması, mevcut bulunulan konumdan istenen hedef konuma kadar tüm kayıtları okuyarak gider (Tape).
- **Direct:** İstlenen konuma doğrudan konumlanır. İstlenen konum okuma/yazma mekanizmasının altına gelene kadar beklenir. Erişim süresi önceki bulunulan konuma bağlıdır (HDD, CD).
- **Random:** İstlenen konuma doğrudan gidilir. Erişim süresi önceki konuma bağlı değildir (Main memory).
- **Associative:** Arama adrese göre değil içeriğe göre yapılır. Aranana veriyle tüm hafıza alanları eşzamanlı karşılaştırılır (Cache).

## Hafıza sistemleri karakteristikleri

### Performance

- **Access time (latency):**
  - RAM için adres bilgisinin verilmesinden verinin alınmasına/yazılmasına başlayıncaya kadar geçen süredir.
  - Diğerleri için okuma/yazma mekanizmasının istenen konuma ulaşması için geçen süredir.
- **Memory cycle time:**
  - RAM için iki erişim süresi arasındaki toplam süredir.
- **Transfer rate:**
  - Veri aktarım hızıdır.
  - RAM için  $(1 / \text{cycle time}) * \text{block size}$  ile ifade edilir.
  - Non-random access memory için  $T_n = T_A + (n / R)$  ile ifade edilir.
  - $T_A$  ortalama erişim süresi;  $n$  bit sayısı ve  $R$  aktarım hızı (bps).
  - $T_n$ ,  $n$  adet bit için okuma veya yazma süresidir.

## Hafıza sistemleri karakteristikleri

### Physical type

- **Semiconductor:**
  - Random access memory'lerde kullanılır.
- **Magnetic:**
  - Disk ve tape ünitelerinde kullanılır.
- **Optical ve magneto-optical:**
  - CD ve DVD'lerde kullanılır.

## Hafıza sistemleri karakteristikleri

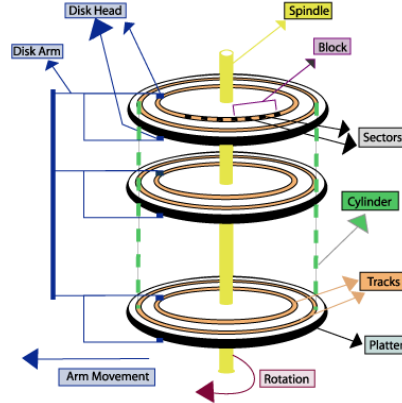
### Physical characteristics

- **Volatile/nonvolatile:**
  - Elektrik kesildiğinde veri kaybolan (semiconductor)
  - Elektrik kesildiğinde veri kaybolmayan (magnetic-surface)
- **Erasable/nonerasable:**
  - İçeriği silinebilen (EEPROM)
  - İçeriği silinemeyen (ROM)

## Hafıza sistemleri karakteristikleri

### Organization

- Word oluşturmak için bitlerin yerleşimini ifade eder (interleaved, sequential).



"Database Management Systems", Ramakrishnan & Gehrke, p. 307 Redrawn by L. Kisinski

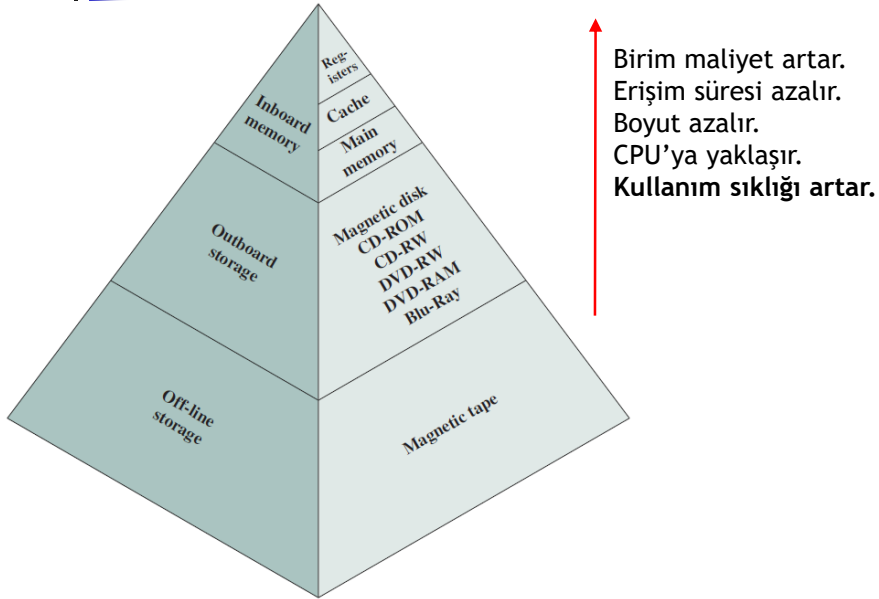
## Konular

- Hafıza sistemleri karakteristikleri
- Hafıza hiyerarşisi**
- Önbellek prensipleri
- Önbellek tasarım bileşenleri
  - Cache size
  - Mapping function
  - Replacement algorithms
  - Write policy
  - Line size
  - Number of caches

## Hafıza hiyerarşisi

- **CPU tarafından** hafıza birimlerine **erişim süresi kısaltıkça, bit başına maliyet artar.**
- **Kapasitesi büyük** hafıza birimlerinin **bit başına maliyeti daha düşüktür.**
- **Kapasitesi büyük** hafıza birimlerine **erişim süresi daha büyüktür.**

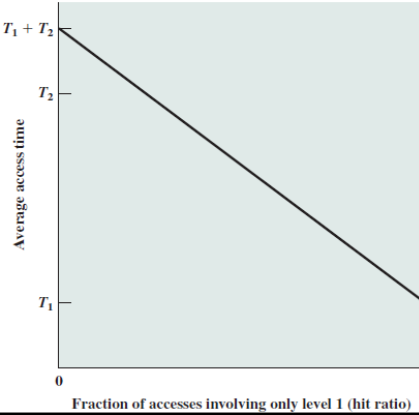
## Hafıza hiyerarşisi



## Hafıza hiyerarşisi

### Örnek:

- Bir CPU iki seviyeli hafızaya erişiyor.
- **Birinci seviye 1000 word** ve **ikinci seviye 100.000 word** kapasitededir.
- Birinci seviyeye **erişim süresi 0,01μs**, ikinci seviyeye **erişim süresi 0,1μs**.
- CPU ilk önce birinci seviyeye, yoksa ikinci seviyeye erişmektedir.
- **$T_1$  birinci seviyeye**  
 **$T_2$  ikinci seviyeye**  
**erişim süresidir.**



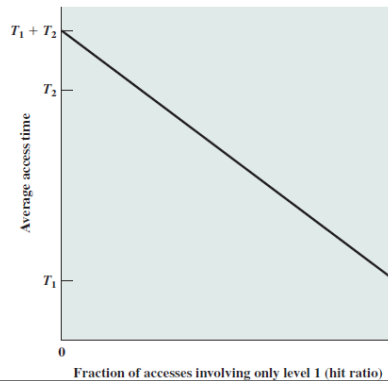
## Hafıza hiyerarşisi

### Örnek:

- Grafikte, verinin birinci seviye hafızada bulunma oranına göre erişim süresi görülmektedir. **Birinci seviye önbellegi ifade eder.**
- Verinin birinci seviyede bulunma oranı %95 ise ortalama erişim süresi (OES) nedir?

$$\begin{aligned} \text{OES} &= (0,95)(0,01\mu\text{s}) + (0,05)(0,01\mu\text{s} + 0,1\mu\text{s}) \\ &= 0,0095 + 0,0055 = 0,015\mu\text{s} \end{aligned}$$

Ortalama erişim süresi **0,01μs**'ye daha yakındır (%95 oranında birinci seviyede bulunmuştur).





## Hafıza hiyerarşisi

CPU'nun erişim sıklığına göre hafıza hiyerarşisi:

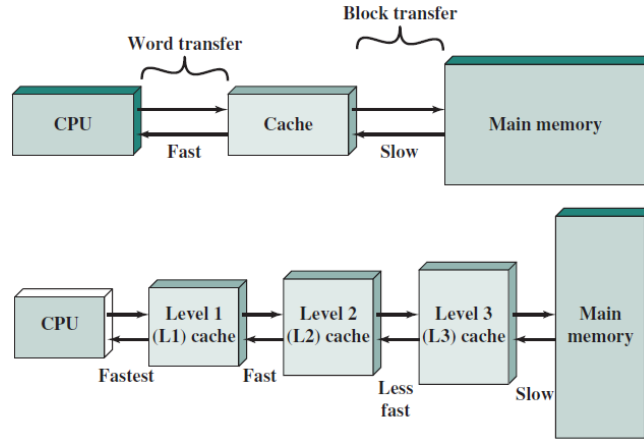
- Registers
- L1 Cache
- L2 Cache
- L3 Cache
- Main memory
- Disk cache
- Disk
- Optical
- Tape

## Konular

- Hafıza sistemleri karakteristikleri
- Hafıza hiyerarşisi
- **Önbellek prensipleri**
- Önbellek tasarım bileşenleri
  - Cache size
  - Mapping function
  - Replacement algorithms
  - Write policy
  - Line size
  - Number of caches

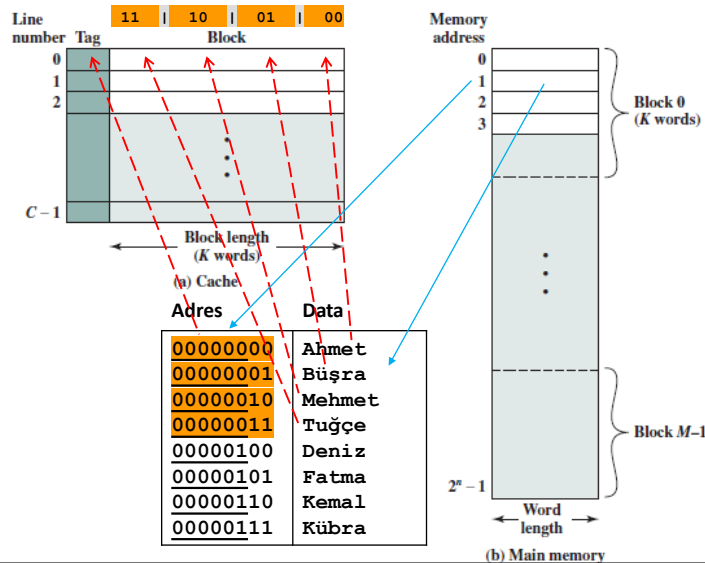
## Önbellek prensipleri

- Önbellek, **main memory ile CPU arasına yerleştirilir.**
- CPU önce önbelleğe erişir, aranan veri yoksa main memory'ye erişir.
- Eğer aranan veri main **memory'de ise** içinde bulunduğu blok ile birlikte alınır. **Önbelleğe ve CPU'ya aktarılır.**



## Önbellek prensipleri

- Önbellek ve main memory yapısı şekilde görülmektedir.



$M$  blok  
Her blok  $K$  word

$$M = 2^n / K$$

$$C \ll M$$

(b) Main memory

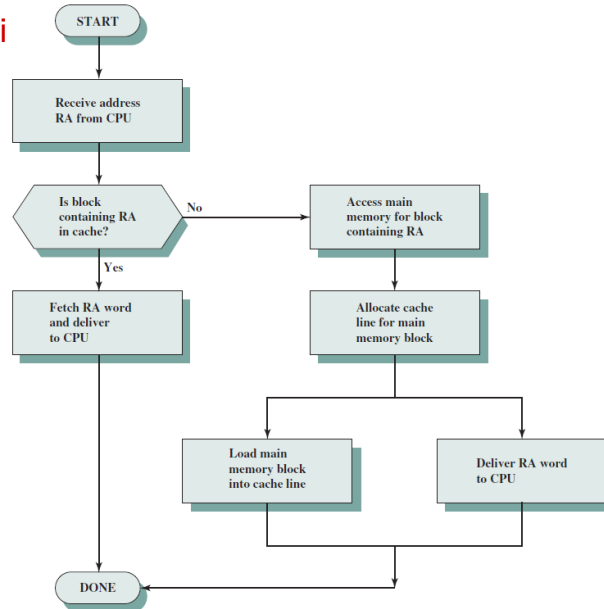
## Önbellek prensipleri

### Önbellek işlemi

- CPU bir adres içeriğini ister.
- Önbelleğe bakılır.
- Önbellekte bulunursa alınır.
- Önbellekte yoksa hafızada içinde bulunduğu blok alınır ve önbelleğe aktarılır.
- Önbellekten CPU'ya aktarılır.

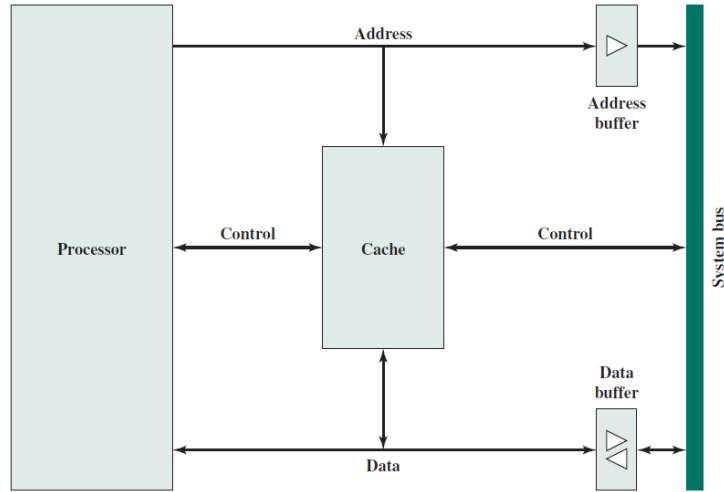
## Önbellek prensipleri

### Önbellek erişimi



## Önbellek prensipleri

### Önbellek organizasyonu



## Konular

- Hafıza sistemleri karakteristikleri
- Hafıza hiyerarşisi
- Önbellek prensipleri
- **Önbellek tasarım bileşenleri**
  - Cache size
  - Mapping function
  - Replacement algorithms
  - Write policy
  - Line size
  - Number of caches

## Önbellek tasarım bileşenleri

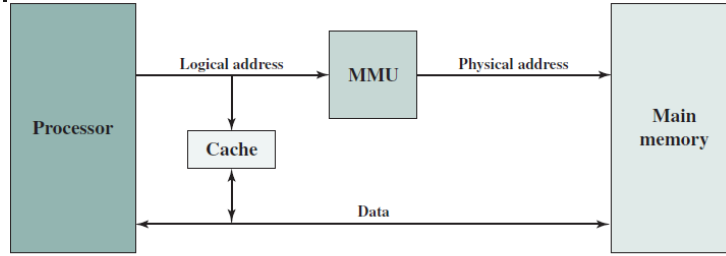
<b>Cache Addresses</b>	<b>Write Policy</b>
Logical	Write through
Physical	Write back
<b>Cache Size</b>	<b>Line Size</b>
<b>Mapping Function</b>	<b>Number of Caches</b>
Direct	Single or two level
Associative	Unified or split
Set associative	
<b>Replacement Algorithm</b>	
Least recently used (LRU)	
First in first out (FIFO)	
Least frequently used (LFU)	
Random	

## Önbellek tasarım bileşenleri

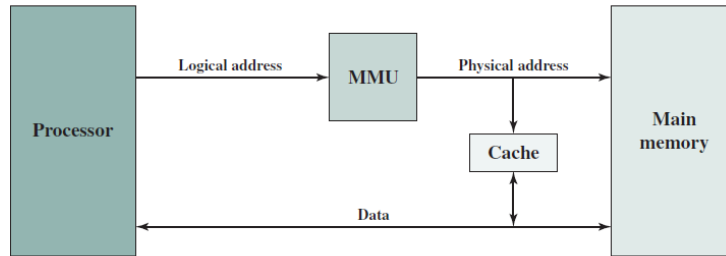
### Cache adres

- Virtual adres kullanıldığında, sistem tasarımcısı **önbelleği memory management unit (MMU) ile CPU arasına** veya **MMU ile main memory arasına** yerleştirebilir.
- Logical cache veriyi virtual adrese göre saklar.
- Fiziksel cache veriyi fiziksel adrese göre saklar.
- Logical cache MMU'ya gidilmeden veriyi sağladığı için daha hızlıdır.

## Önbellek tasarım bileşenleri



(a) Logical cache



(b) Physical cache

## Önbellek tasarım bileşenleri

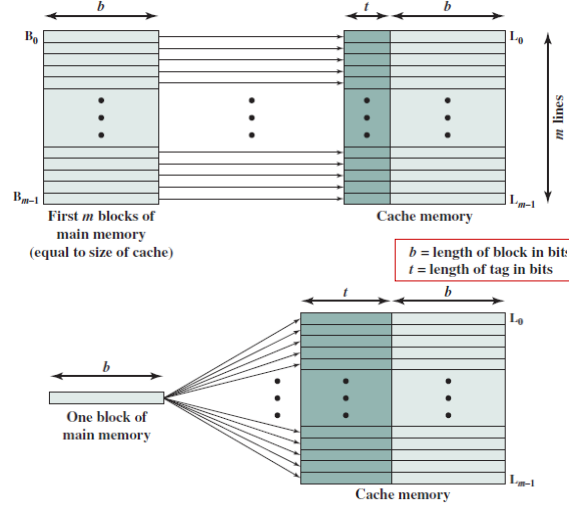
### Mapping function

- **Önbellekteki satır sayısı main memory'den çok az** olduğu için eşleştirme fonksiyonu kullanılarak aktarma yapılır.
- Mapping function **hafızadaki bir bloğun önbelleğe nasıl yerleştirileceğini belirler.**
- **Direct, full associative** ve **set associative** olarak üç yöntem kullanılır.

## Önbellek tasarım bileşenleri

### Mapping function

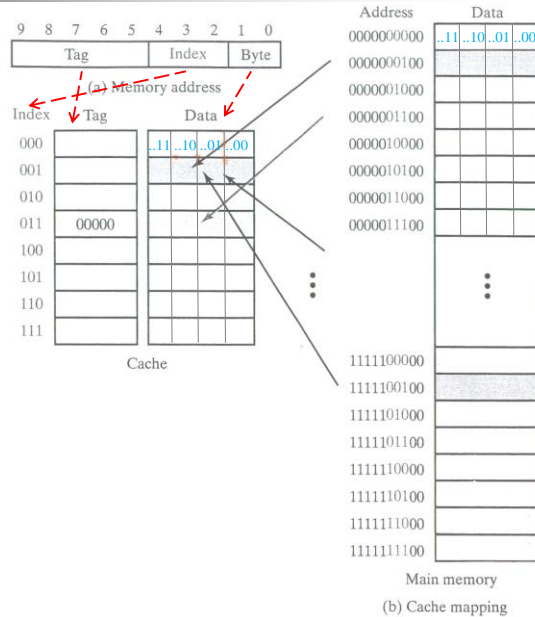
- Direct mapping ve associative mapping

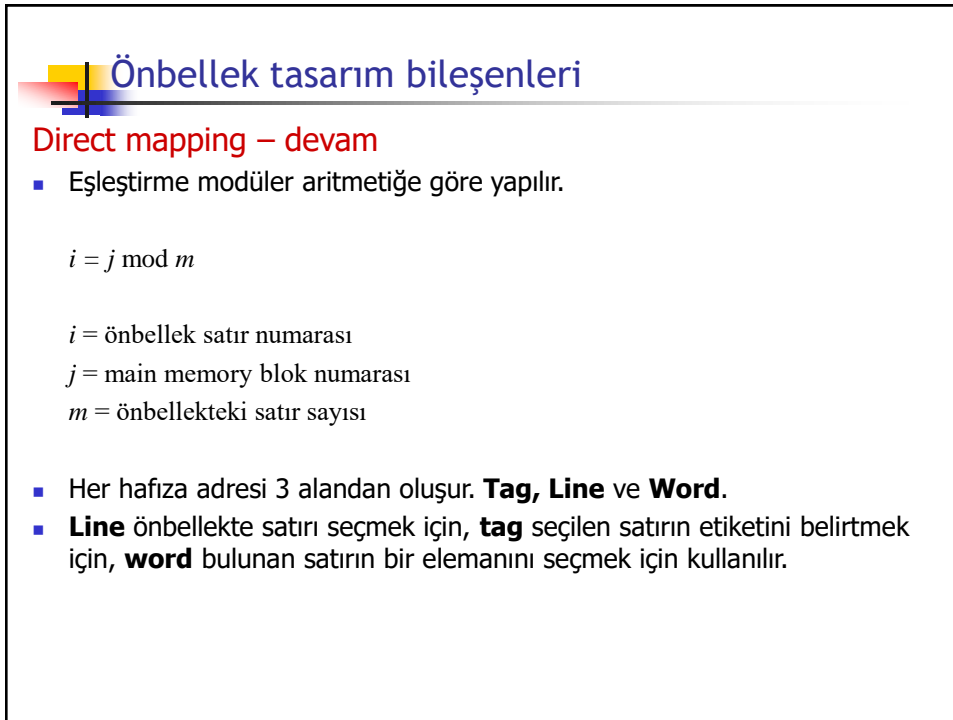
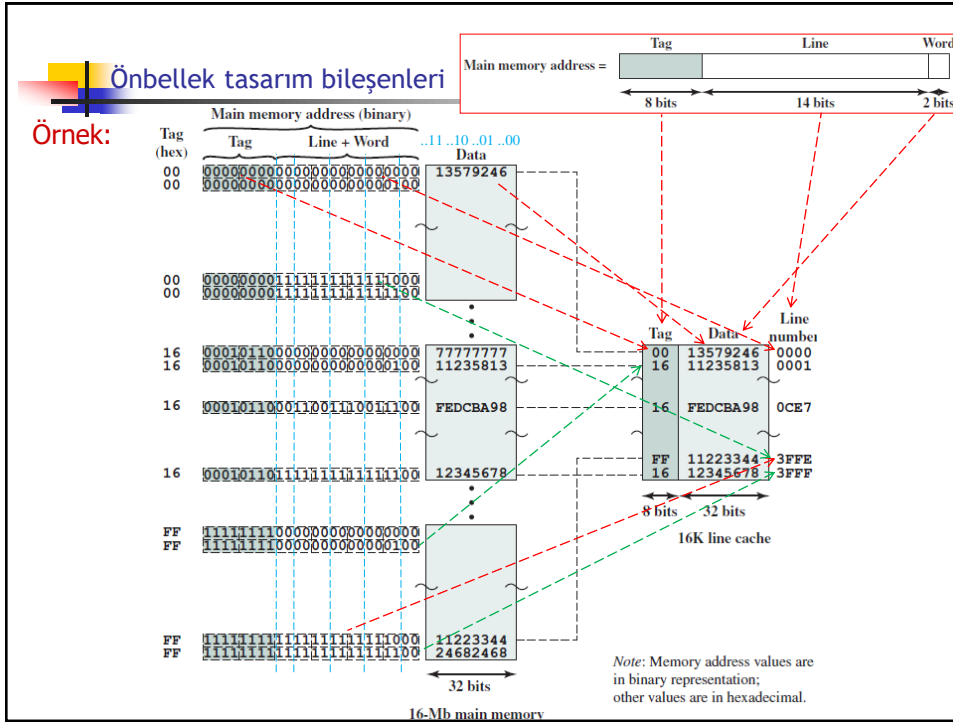


## Önbellek tasarım bileşenleri

### Direct mapping

- Şekilde **önbellek satır boyutu 32-bit** alınmıştır.
- Hafızada **her blok 4-byte** kapasitedir.
- Önbellekte **satır adresleme 3-bit** ile yapılmaktadır.
- Hafızadaki **256 blok** önbellekteki toplam **8 satıra** eşleştirilmektedir.
- Önbellekteki her satırı **32 blok** paylaşmaktadır.

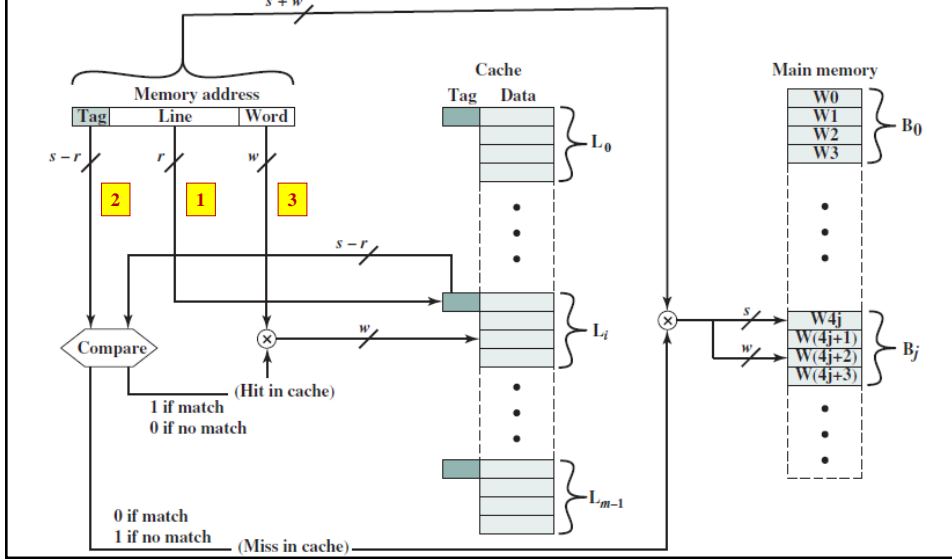






## Önbellek tasarım bileşenleri

### Direct mapping – devam



## Önbellek tasarım bileşenleri

### Direct mapping – devam

- Adres boyutu =  $(s+w)$  bit
- Adreslenebilir alan sayısı =  $2^{s+w}$  word
- Blok boyutu = satır boyutu =  $2^w$  word
- Hafızadaki blok sayısı =  $(2^{s+w}) / (2^w) = 2^s$  word
- Önbellekteki satır sayısı =  $2^r$
- Önbellek boyutu =  $2^{r+w}$
- Tag boyutu =  $(s-r)$  bit

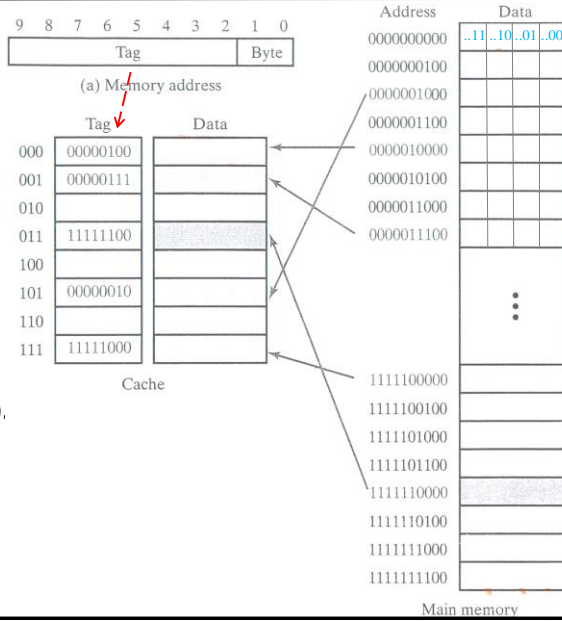
### Avantaj / dezavantaj

- Oluşturmak **basit ve ucuzdur**.
- **Bir blok sadece bir satıra yazılabilir.**
- Aynı satıra eşleşen iki blok sürekli çalıştığında performans düşer.

## Önbellek tasarım bileşenleri

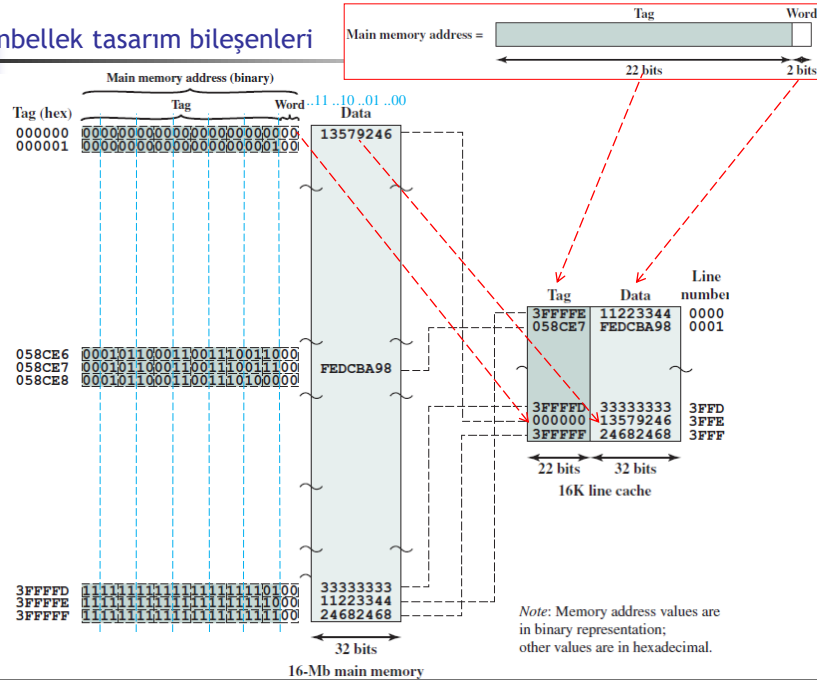
### Full associative mapping

- Direct mapping'teki bir bloğun sadece bir satıra eşleştirilmesi dezavantajı ortadan kaldırılır.
- **Bir blok önbellekte istenilen satıra eşleştirilir.**
- Aranan bilginin önbellekte olup olmadığı tüm satırlarda eşzamanlı kontrol edilir (içerik arama).
- Hafızadan alınan bloğun önbellekte hangi satıra yazılacağı **replacement algoritması** ile belirlenir.



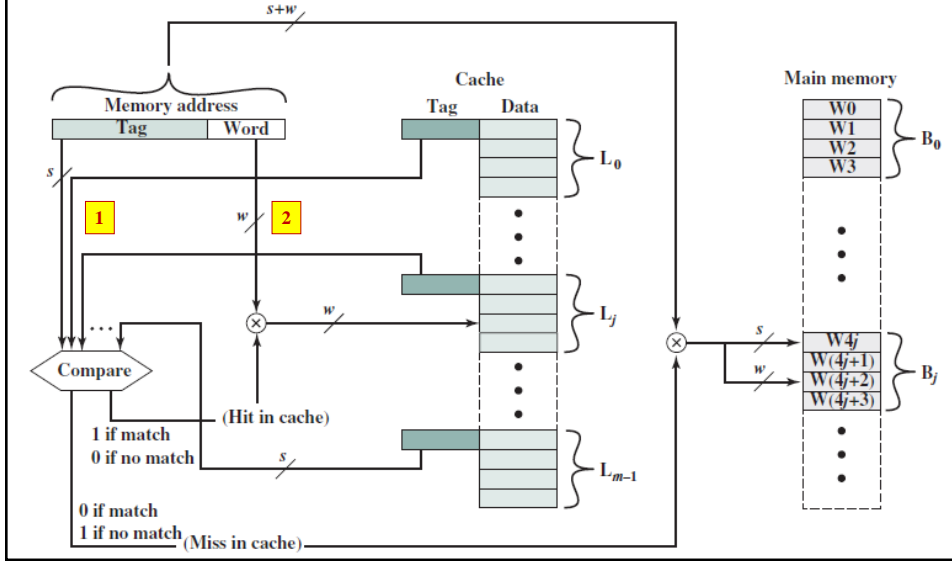
## Önbellek tasarım bileşenleri

Örnek:



## Önbellek tasarım bileşenleri

### Full associative mapping – devam



## Önbellek tasarım bileşenleri

### Full associative mapping – devam

- Adres boyutu =  $(s+w)$  bit
- Adreslenebilir alan sayısı =  $2^{s+w}$  word
- Blok boyutu = satır boyutu =  $2^w$  word
- Hafızadaki blok sayısı =  $(2^{s+w}) / (2^w) = 2^s$  word
- Önbellekteki satır sayısı = tanımlı değil
- Tag boyutu =  $s$  bit

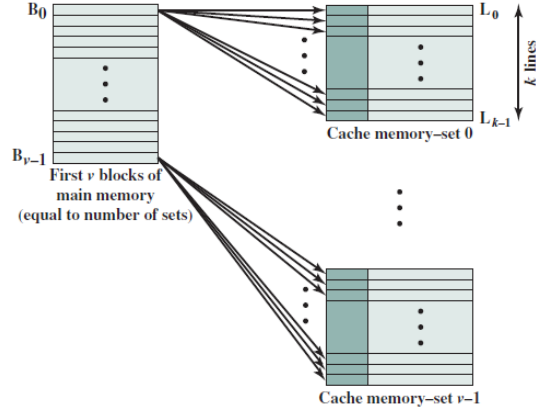
### Avantaj / dezavantaj

- Yapısı karmaşıktır.
- Bir blok uygun olan bir satıra yazılabilir.
- Önbellekte eşzamanlı arama hızı düşüktür.

## Önbellek tasarım bileşenleri

### Set associative mapping

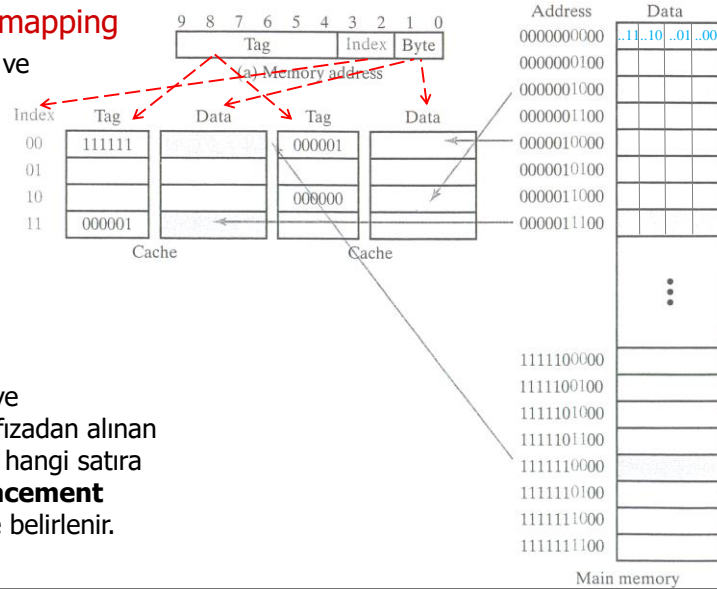
- Önbellek satırları birden fazla blok ile oluşturulur.



## Önbellek tasarım bileşenleri

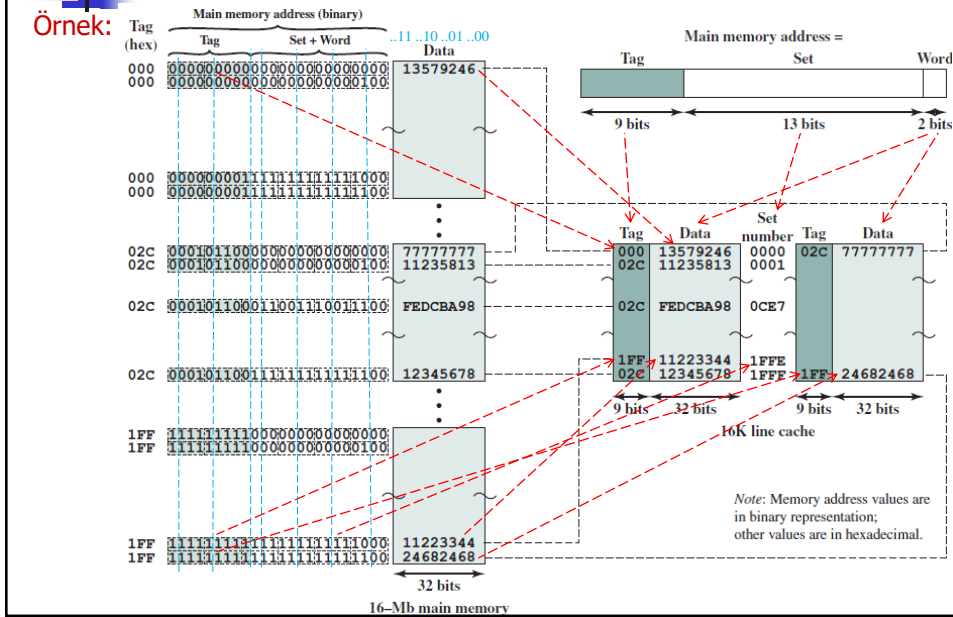
### Set associative mapping

- Direct mapping ve associative mapping birleşimidir.
- Önbellek belirli sayıda kümeden oluşur.**
- Her küme kendi içinde associative yapıdadır ve hafızadan alınan bloğun kümede hangi satıra yazılacağı **replacement algoritması** ile belirlenir.



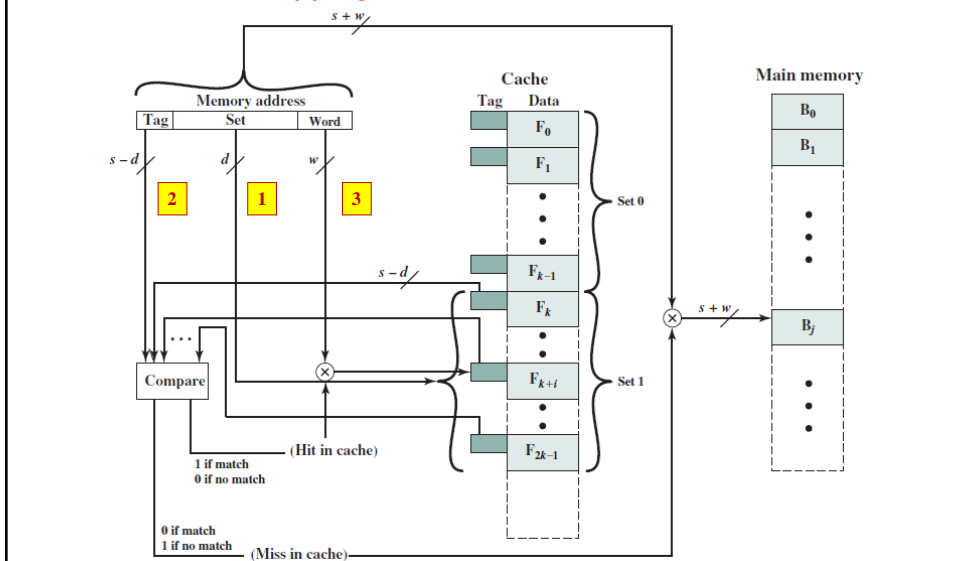
## Önbellek tasarım bileşenleri

Örnek:



## Önbellek tasarım bileşenleri

Set associative mapping – devam



## Önbellek tasarım bileşenleri

### Set associative mapping– devam

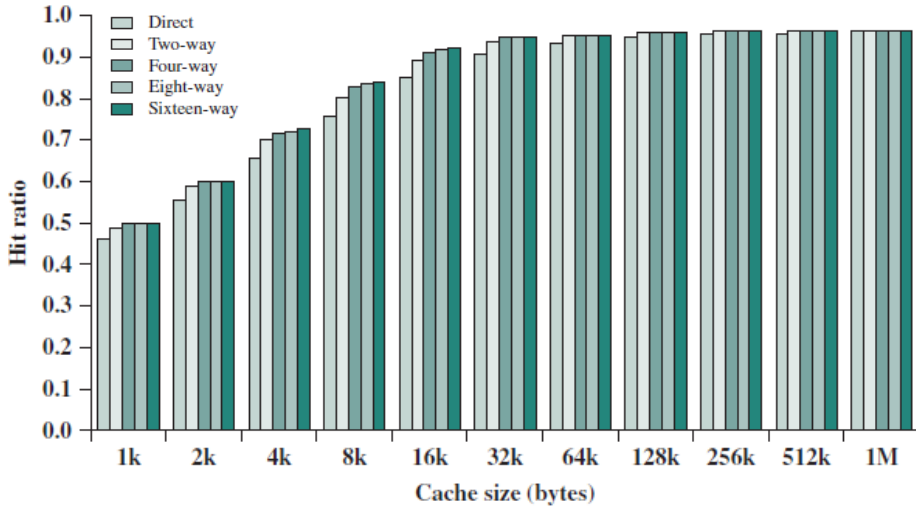
- Adres boyutu =  $(s+w)$  bit
- Adreslenebilir alan sayısı =  $2^{s+w}$  word
- Blok boyutu = satır boyutu =  $2^w$  word
- Hafızadaki blok sayısı =  $(2^{s+w}) / (2^w) = 2^s$  word
- Kümedeki satır sayısı =  $k$
- Küme sayısı =  $2^d$
- Önbellek boyutu =  $k \cdot 2^d$
- Tag boyutu =  $(s-d)$  bit

### Avantaj / dezavantaj

- Yapısı direct mapping'e göre karmaşıktır.
- Bir blok sadece kendisine ait bir kümedeki istenilen satıra yazılabilir.
- Önbellekte küme içinde eşzamanlı arama hızı düşüktür.

## Önbellek tasarım bileşenleri

### Cache boyutuna göre hit ratio (direct ve set associative)



## Önbellek tasarım bileşenleri

### Replacement algorithms

- **Direct mapping'te** sadece bir satır seçilebildiği için replacement algoritması kullanılmaz.
- **Full associative** ve **set associative** eşleştirmede replacement algoritmaları kullanılır.
- **Least recently used (LRU):**
  - En uzun süre kullanılmayan satıra yazılır.
  - Her satır için USE field kullanılır.
- **First in first out (FIFO):**
  - İlk gelen satıra yazılır.
- **Least frequently used (LFU):**
  - En az sıklıkta kullanılan satıra yazılır.
  - Her satır için counter kullanılır.
- **Random:**
  - Rastgele bir satır seçilir ve o satıra yazılır.

## Önbellek tasarım bileşenleri

### Write policy

- Önbellekteki **veri değiştiyse** üzerine veri yazılmadan **hafızaya aktarılmalıdır**.
- Eğer hafıza birden fazla CPU tarafından ortak kullanılıyorsa, önbellekteki güncel verinin hafızaya aktarılma yöntemi önemlidir.
- **Write through:** Önbellekteki her yazma işlemi doğrudan hafızaya da aktarılır.
- Çok işlemcili sistemlerde bus sürekli izlenir ve değişiklikler aktarılır.
- Bus üzerindeki trafik yüksektir.
- **Write back:** Önbellekteki veri değiştiği anda değil, önbellekten atılacağı anda hafızaya yazılır. Her satır için UPDATE field kullanılır.
- Bus trafiği write through'a göre daha düşüktür.
- **Write once:** Önbellekteki ilk değişiklikler hemen yazılır (write through), diğerleri atılırken yazılır (write back).

## Önbellek tasarım bileşenleri

### Line size

- Önbellekte bir veri bulunamadığında, hafızadan sadece o veri değil ait olduğu blok alınır.
- Satır boyutu arttıkça **başlangıçta hit ratio artar** daha **sonra düşmeye başlar**.
- Blok boyutu arttıkça **yakın zamanda kullanılmayacak verilerde alınmaya başlar**.

## Önbellek tasarım bileşenleri

### Number of caches

- Önbellek çok seviyeli ve veri ile komut için ayrı ayrı oluşturulabilir.

### Multilevel caches

- On-chip ve off-chip olarak oluşturulabilir.
- L1, L2 ve L3 şeklinde üç seviyeli kullanımı vardır.
- On-chip önbellek CPU'nun external bus trafiğini azaltır.

### Unified/split caches

- Split önbelleklerde **komut ve data** için ayrı ayrı kısımlar kullanılır.
- Unified önbelleklerde **hit oranı yüksektir**. Çünkü komut ve data arasındaki fetch yoğunluğuna göre kendini update eder.
- Unified önbelleklerde tek kısım olduğundan **yapısı basittir**.
- Split önbelleğin en önemli avantajı, instruction cycle'da **instruction fetch/decode unit** ile **execution unit**'i bağımsız hale getirmesidir. **Pipelining performansı için önemlidir**.



## Ödev

- Virtual memory kullanıldığında uygulamalar arasında geçişte virtual cache ile karşılaşılan problemler ve kullanılan çözümler hakkında detaylı bir araştırma ödevi hazırlayınız.