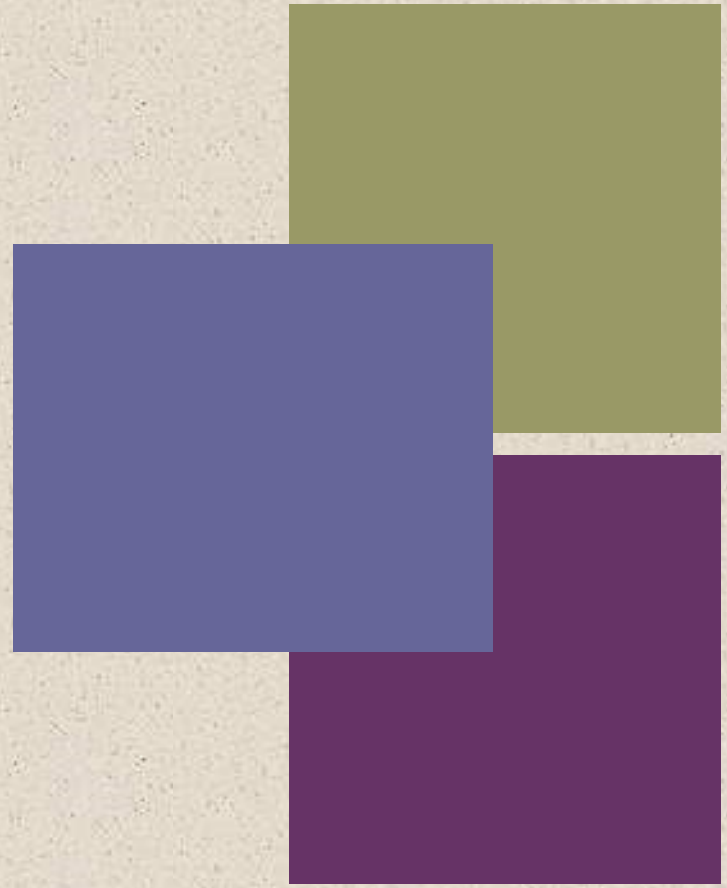
William Stallings Computer Organization and Architecture 10th Edition

Orijinal slaytların çevirisidir.



Chapter 4

Cache Memory

Bilgisayar belleğinin karmaşık konusu, bellek sistemlerini temel özelliklerine göre sınıflandırılırsa daha yönetilebilir hale gelir.

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

Table 4.1
Key Characteristics of Computer Memory Systems

Bilgisayar Bellek Sistemlerinin Önemli Karakteristikleri

Characteristics of Memory Systems

- Konum (*Location*)
  - Belleğin bilgisayarda dahili (*internal*) mi yoksa harici (*external*) mi olduğunu ifade eder
  - Dahili bellek genellikle ana bellek ile eşdeğerdir. Fakat dahili belleğin daha farklı formları vardır.
    - İşlemci, register biçiminde kendi yerel belleğine ihtiyaç duyar
    - Önbellek (*Cache*) başka bir dahili bellek biçimidir
  - Harici bellek, işlemci tarafından I/O denetleyicileri aracılığıyla erişilebilen çevresel depolama aygıtlarından oluşur
- Kapasite (*Capacity*)
  - Bellek tipik olarak bayt cinsinden ifade edilir
- Transfer birimi (*Unit of transfer*)
  - Dahili bellek için aktarım birimi, bellek modülüne giren ve çıkan elektrik hatlarının sayısına eşittir
    - Bu, kelime uzunluğuna eşit olabilir, ancak genellikle 64, 128 veya 256 bit gibi daha büyüktür.

Method of Accessing Units of Data

Bellek türleri arasındaki diğer bir ayrım, veri birimlerine erişim yöntemidir. Bunlar aşağıdakileri içerir:

Sequential access

- Bellek, kayıt (*records*) adı verilen veri birimleri halinde düzenlenir
- Erişim belirli bir doğrusal sırayla yapılmalıdır
- Erişim süresi değişkendir

Ex.:Tape

Direct access

- Paylaşılan bir okuma-yazma mekanizmasını içerir
- Ayrı blokların veya kayıtların fiziksel konuma göre benzersiz bir adresi vardır
- Erişim süresi değişkendir

Access is accomplished by direct Access to reach a general vicinity plus sequential searching, counting, or waiting to reach the final location.

Ex.:Disk

Random access

- Bellekteki her adreslenebilir konum, benzersiz, fiziksel olarak kablolu adresleme mekanizmasına sahiptir.
- Belirli bir konuma erişim süresi, önceki erişimlerin sırasından bağımsızdır ve sabittir.
- Herhangi bir konum rastgele seçilebilir ve doğrudan adreslenebilir ve erişilebilir
- Ana bellek ve bazı önbellek sistemleri rastgele erişimdir

Associative

- Bir kelime (*word*), adresi yerine içeriğinin bir kısmına göre alınır (retrieve)
- Her konumun kendi adresleme mekanizması vardır ve erişim süresi, konumdan veya önceki erişim modellerinden bağımsız olarak sabittir.
- Önbellekler çağrışımlı erişim kullanabilir

Capacity and Performance:

Belleğin en önemli iki özelliği

Kullanıcı bakış açısından, belleğin en önemli iki özelliği kapasite ve performanstır.

Üç **performans** parametresi kullanılır:

Access time (latency)

- Rastgele erişimli bellek için, bir okuma veya yazma işlemi gerçekleştirmek için gereken süredir.
- Rasgele erişimli olmayan bellek için okuma-yazma mekanizmasını istenen konuma konumlandırmak için gereken süredir.

Memory cycle time

- Erişim süresi artı ikinci erişimin başlayabilmesi için gereken ek süre
- Geçici akımların sinyal hatlarında sona ermesi veya yıkıcı bir şekilde okundukları takdirde verileri yeniden oluşturmaları için ek süre gerekebilir.
- İşlemci ile değil, sistem veriyolu (*system bus*) ile ilgili

Transfer rate

- Verilerin bir bellek birimine veya bellek biriminden dışarıya aktarılabilme hızı
- Rastgele erişimli bellek için **1/(cycle time)** değerine eşittir

Çeşitli fiziksel bellek türleri kullanılmıştır.

Memory

- En yaygın biçimler şunlardır:
  - Semiconductor memory
  - Magnetic surface memory
  - Optical
  - Magneto-optical
- Veri depolamanın çeşitli fiziksel özellikleri önemlidir:
  - Geçici/uçucu bellek (*Volatile memory*)
    - Elektrik gücü kapatıldığında bilgi doğal olarak azalır veya kaybolur
  - Kalıcı bellek (*Nonvolatile memory*)
    - Kaydedildikten sonra bilgiler, kasıtlı olarak değiştirilene kadar bozulmadan kalır
    - Bilgileri tutmak için elektrik gücüne gerek yoktur
  - Magnetic-surface memories (Manyetik yüzey bellekleri)
    - Kalıcıdır (*nonvolatile*)
  - Yarı iletken bellek (*Semiconductor memory*)
    - Geçici veya kalıcı olabilir (*volatile or nonvolatile*)
  - Silinemez bellek
    - Depolama biriminin tahrip edilmesi dışında değiştirilemez
    - Bu türdeki yarı iletken bellek salt okunur bellek (ROM) olarak
- Rastgele erişimli bellek için organizasyon önemli bir tasarım meselesidir
  - Organizasyon, kelimeleri oluşturmak için bitlerin fiziksel düzenlemesini ifade eder

Memory Hierarchy

- Bir bilgisayarın belleğindeki tasarım kısıtlamaları üç soruyla özetlenebilir:
  - Ne kadar büyüklükte, ne kadar hızlı, ne kadar pahalı
- Kapasite, erişim süresi ve maliyet arasında bir denge vardır
  - Daha hızlı erişim süresi, bit başına daha yüksek maliyet
  - Daha fazla kapasite, bit başına daha düşük maliyet
  - Daha fazla kapasite, daha yavaş erişim süresi
- Bellek ikileminden çıkış yolu, tek bir bellek bileşenine veya teknolojisine güvenmek değil, bir bellek hiyerarşisi kullanmaktır.

Tipik bir hiyerarşi Şekil 4.1'de gösterilmektedir. Hiyerarşide aşağı inildikçe, aşağıdakiler gerçekleşir:

a. Azalan bit başına düşen maliyet
b. Artan kapasite
c. Artan erişim süresi
d. İşlemci tarafından belleğe erişim sıklığının azalması

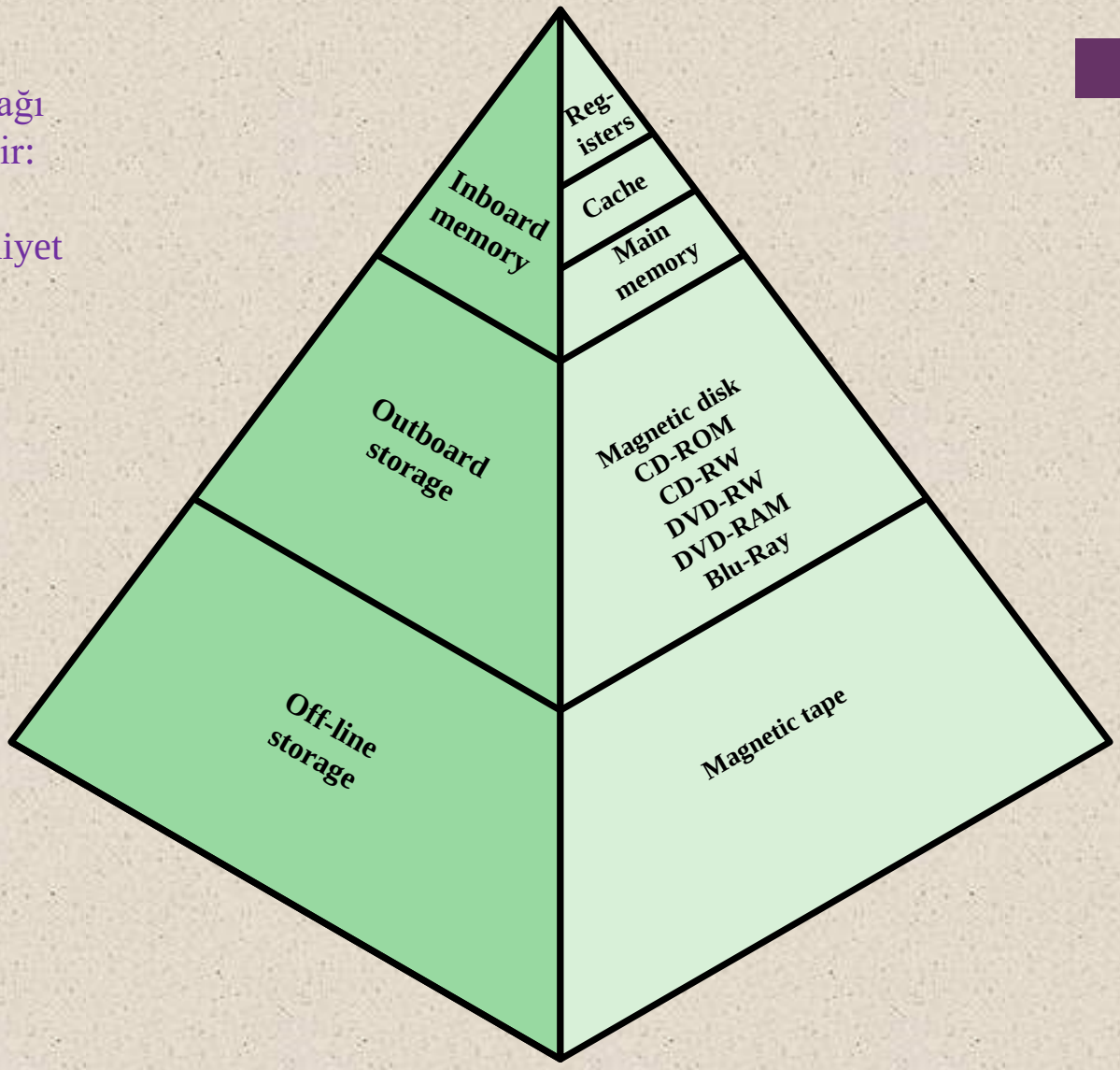
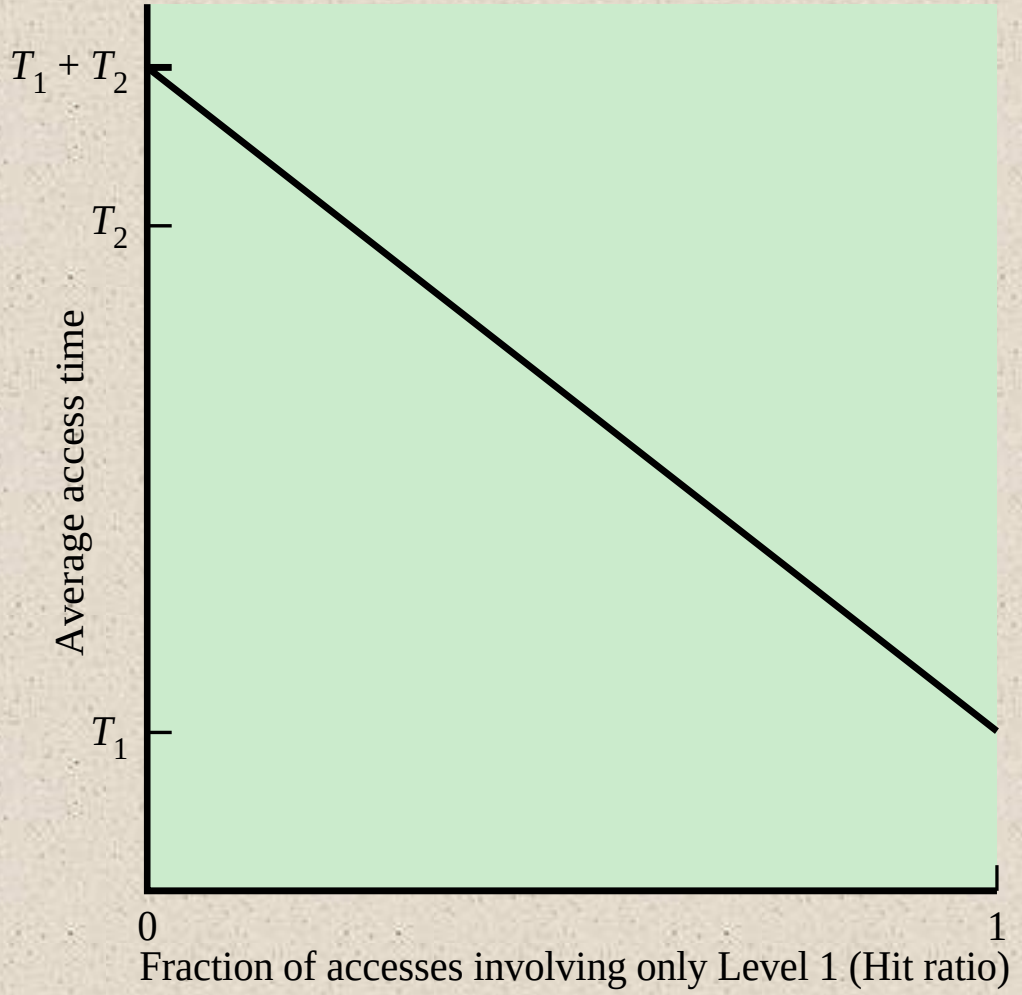


Figure 4.1 The Memory Hierarchy



İşlemcinin iki seviyeli belleğe erişimi olduğunu varsayalım. **Level1** , 1000 kelime içerir ve **0,01 µs** erişim süresine sahiptir; **Level2**, 100.000 kelime içerir ve **0,1 µs** erişim süresine sahiptir. Erişilecek bir kelime Level1'de ise, işlemcinin ona doğrudan eriştiğini varsayın. Level2'de ise, kelime önce L1'e aktarılır ve ardından işlemci tarafından erişilir. Basit olması için, işlemcinin kelimenin L1'de mi yoksa L2'de mi olduğunu belirlemesi için gereken süreyi göz ardı ediyoruz. Şekil 4.2, bu durumu kapsayan eğrinin genel şeklini göstermektedir. Şekil, **H** isabet oranının (*hit ratio*) bir fonksiyonu olarak iki seviyeli bir belleğe ortalama erişim süresini gösterir; burada H, daha hızlı bellekte (örneğin, önbellekte) bulunan tüm bellek erişimlerinin oranı olarak tanımlanır, **T1**, 1. seviyeye erişim süresi ve **T2** 2. seviyeye erişim süresidir. Görüldüğü gibi, Level1 erişimin yüksek yüzdeleri için, ortalama toplam erişim süresi Level2'ye göre 1. seviyeye çok daha yakındır.

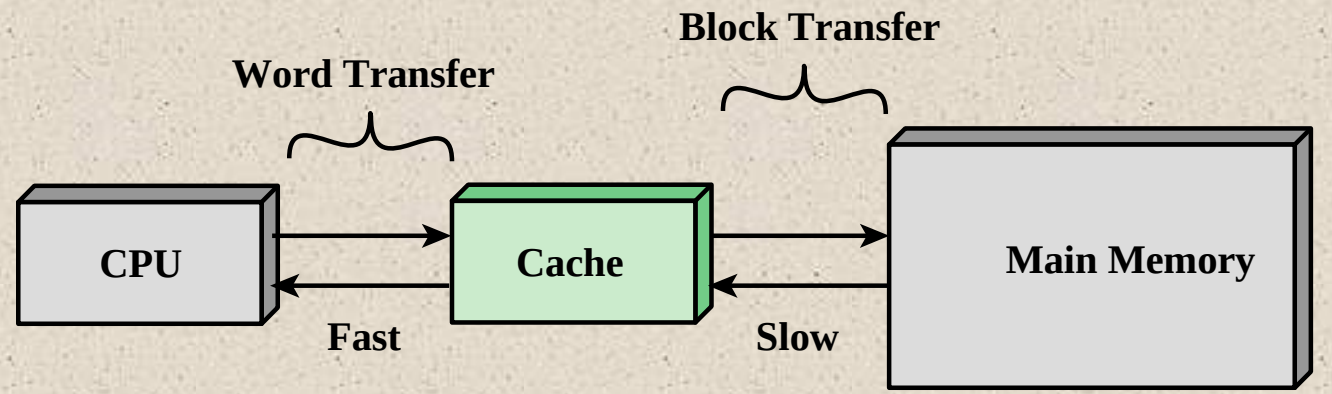
Örneğimizde, bellek erişimlerinin% 95'inin Level 1'de bulunduğunu varsayalım. Buna göre, bir kelimeye erişim için ortalama süre şu şekilde ifade edilebilir:
(0,95) (0,01 µs) + (0,05) (0,01 µs + 0,1 µs) = 0,0095 + 0,0055 = 0,015 µs

Ortalama erişim süresi, istendiği gibi 0.01 µs'ye 0.1 µs'den çok daha yakındır.

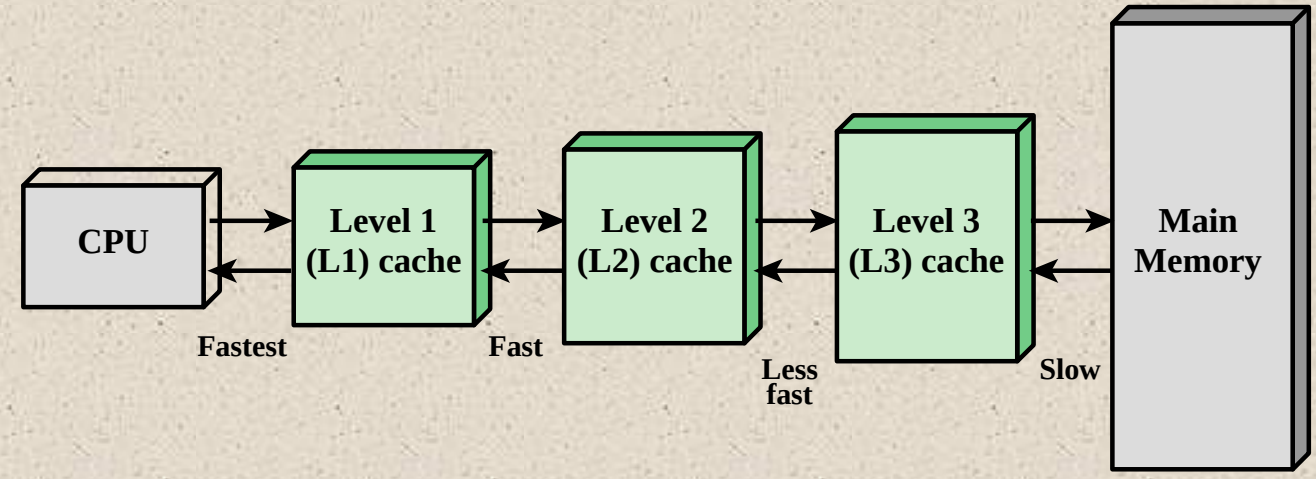
Figure 4.2 Performance of a Simple Two-Level Memory

Memory

- Üç seviyenin kullanılması, yarı iletken belleğin hız ve maliyet açısından farklılık gösteren çeşitli türlerde geldiği gerçeğinden yararlanır.
- Veriler, harici yığın depolama cihazlarında daha kalıcı olarak depolanır
- Harici, kalıcı bellek; ikincil bellek (***secondary*** *memory*) veya yardımcı bellek (***auxiliary*** *memory*) olarak da adlandırılır
- Disk önbelleği (Disk cache)
  - Ana belleğin bir kısmı, diske okunacak verileri geçici olarak tutmak için bir arabellek olarak kullanılabilir.
  - Çok sayıda küçük veri aktarımı yerine birkaç büyük veri aktarımı amacıyla kullanılabilir
  - Veriler, diskten yavaş bir şekilde değil, yazılım önbelleğinden (*software cache*) hızlı bir şekilde alınabilir



(a) Single cache



(b) Three-level cache organization

Figure 4.3 Cache and Main Memory

Önbellek; pahalı, yüksek hızlı belleğin bellek erişim süresini, daha ucuz, daha düşük hızlı belleğin büyük bellek boyutu ile birleştirmek için tasarlanmıştır.

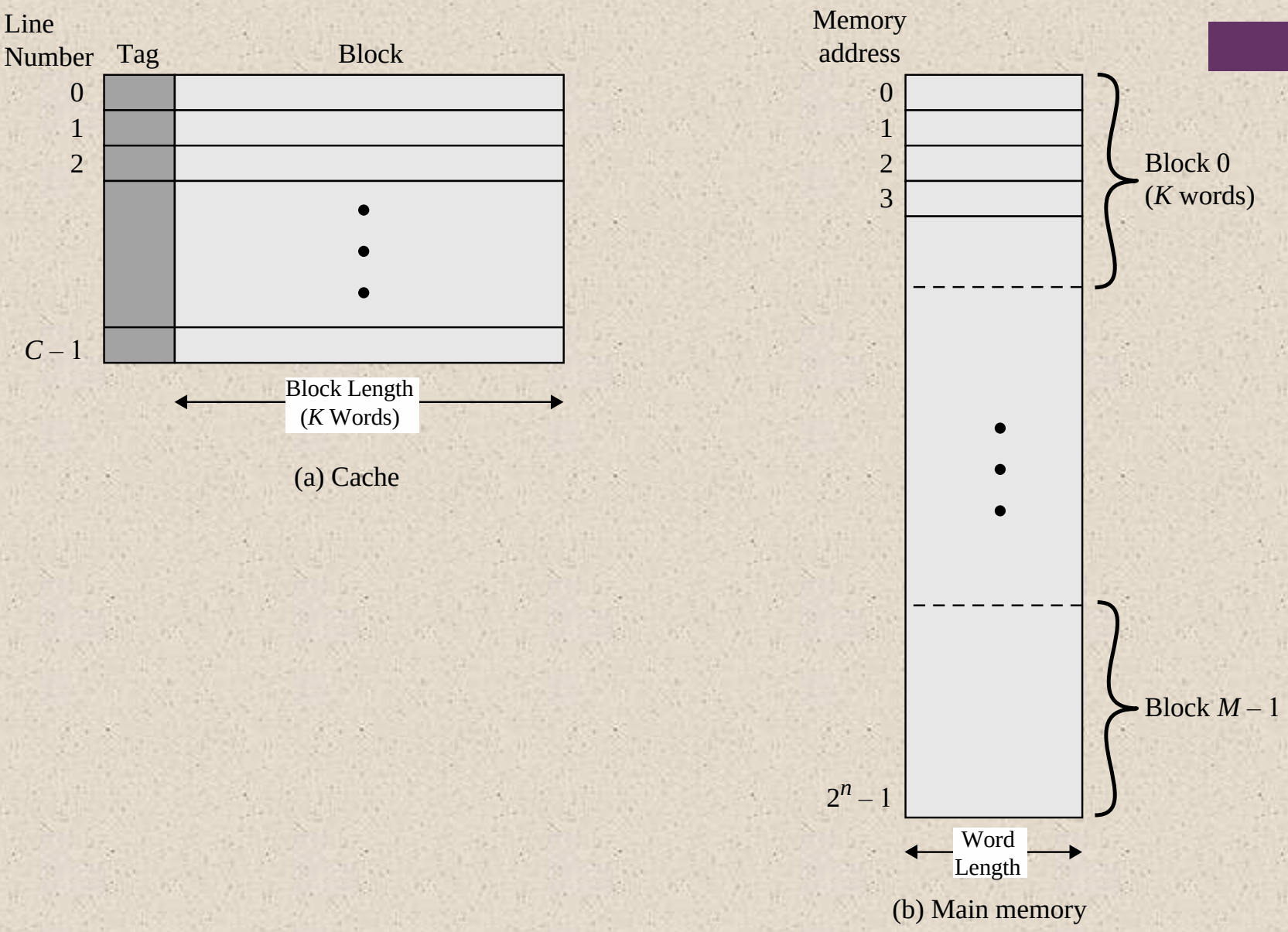
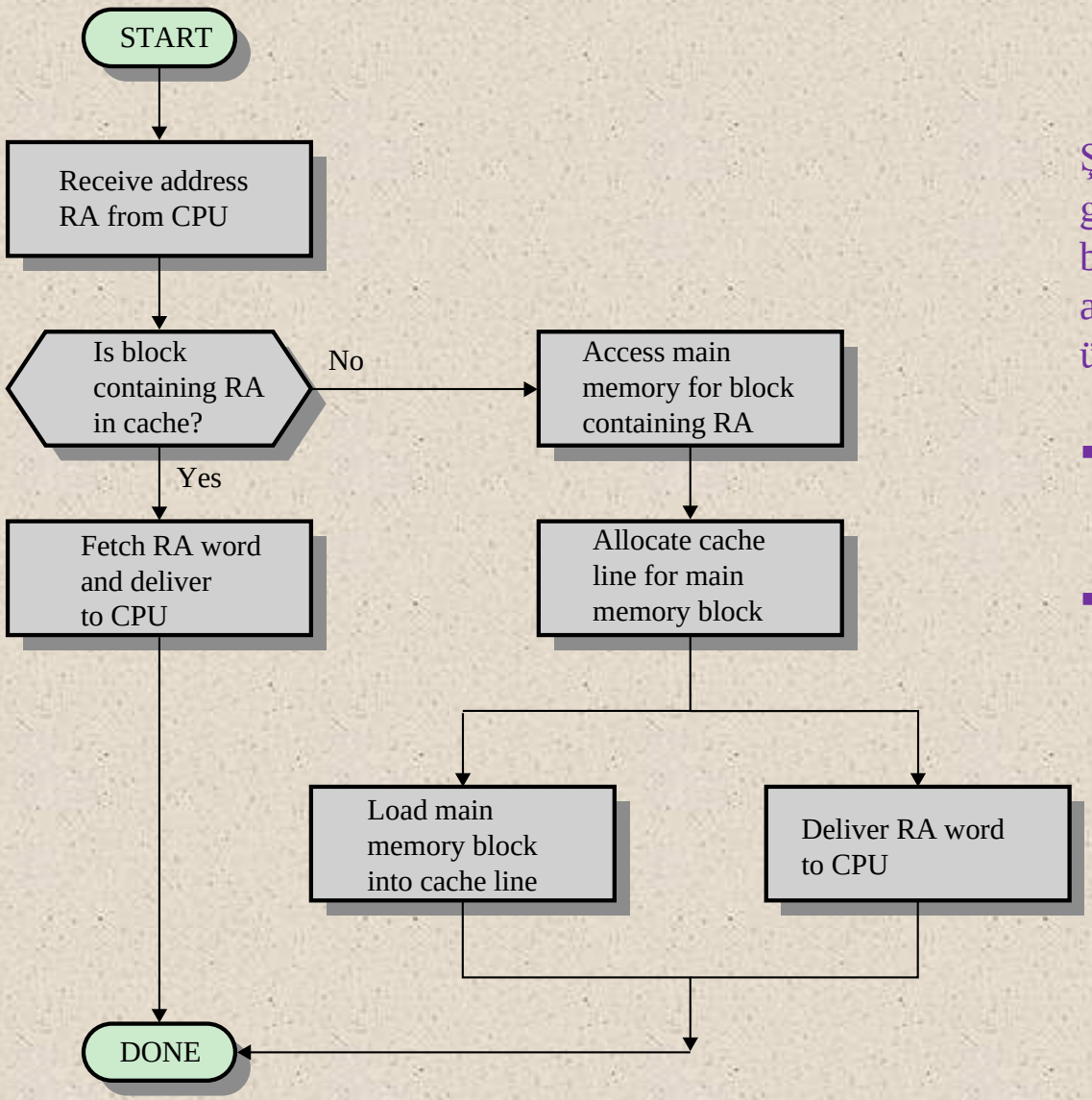


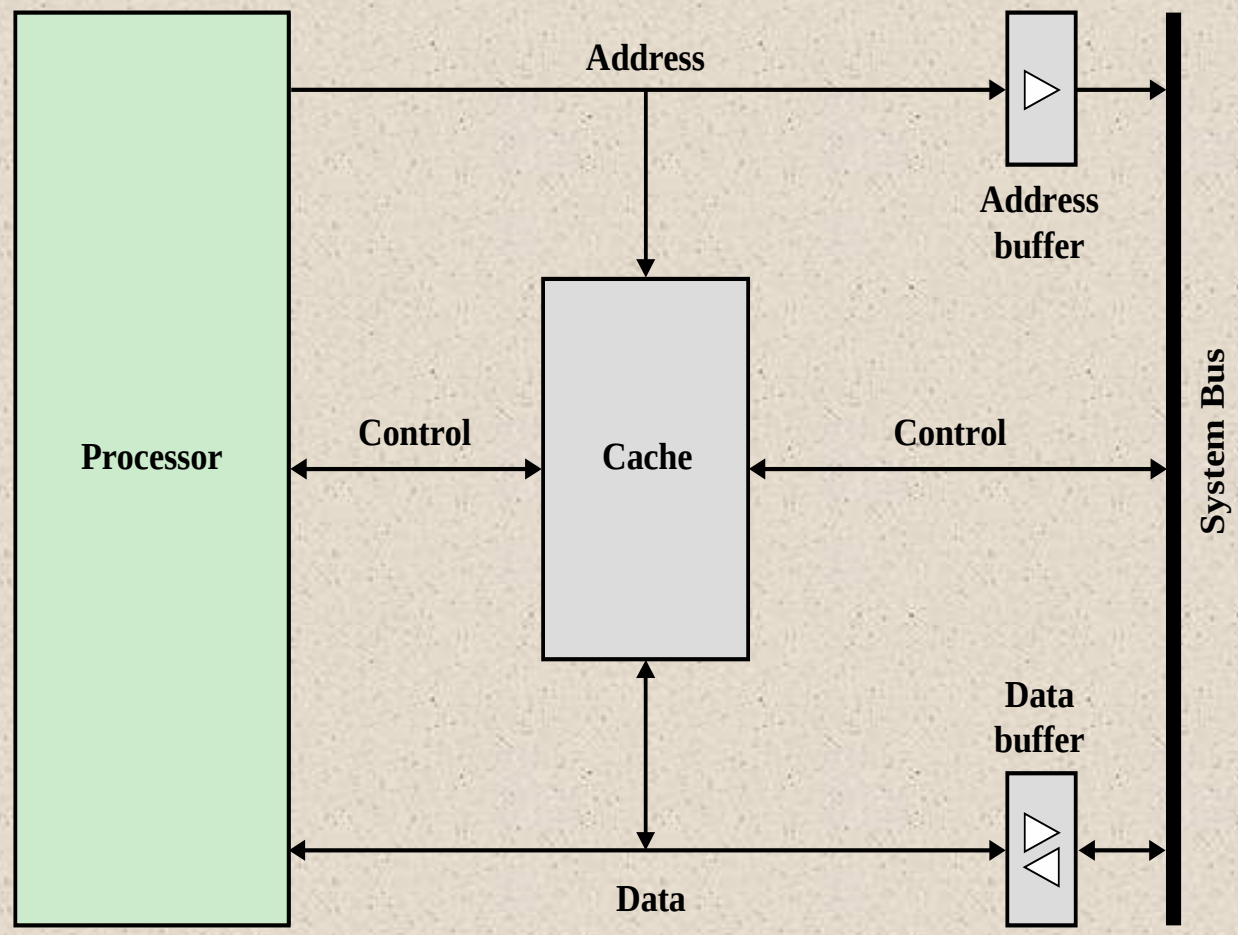
Figure 4.4 Cache/Main-Memory Structure



Şekil 4.5, okuma işlemini göstermektedir. İşlemci, okunacak bir kelimenin (*word*) okuma adresini (**read address-RA**) üretir.

- Kelime önbellekte bulunuyorsa işlemciye teslim edilir.
- Aksi takdirde, bu kelimeyi içeren blok önbelleğe yüklenir ve kelime, işlemciye teslim edilir.

Figure 4.5 Cache Read Operation



Bu organizasyonda, önbellek işlemciye veri, kontrol ve adres hatları aracılığıyla bağlanır. Veri ve adres hatları ayrıca, ana belleğe erişilen bir sistem veri yoluna bağlanan veri ve adres arabelleklerine (*data and address buffers*) de eklenir. Bir **önbellek isabeti** (***cache hit***) gerçekleştiğinde, veri ve adres buffer'ları devre dışı bırakılır ve iletişim, sistem veriyolu trafiği olmadan **yalnızca işlemci ve önbellek arasındadır**.

Bir **önbellek isabetsizliği** (***cache miss***) meydana geldiğinde, istenen adres sistem veriyoluna yüklenir ve veriler, veri buffer'ından hem önbelleğe hem de işlemciye geri gönderilir. Diğer organizasyonlarda, önbellek fiziksel olarak tüm veri, adres ve kontrol hatları için işlemci ile ana bellek arasına yerleştirilir.

Figure 4.6 Typical Cache Organization

Bu ikinci durumda, bir önbellek isabetsizliği durumu için, istenen kelime önce önbelleğe okunur ve ardından önbellekten işlemciye aktarılır.

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

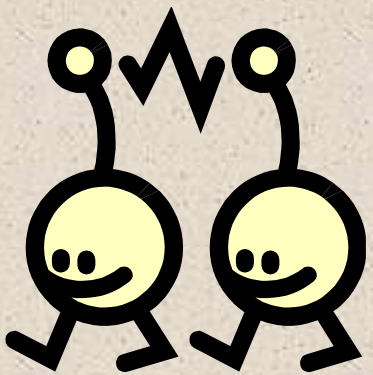
Table 4.2
Elements of Cache Design

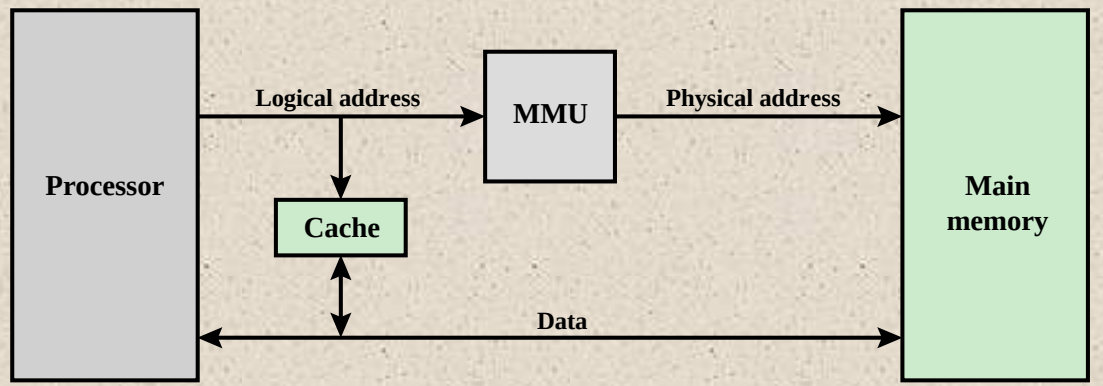
Cache Addresses

Virtual Memory

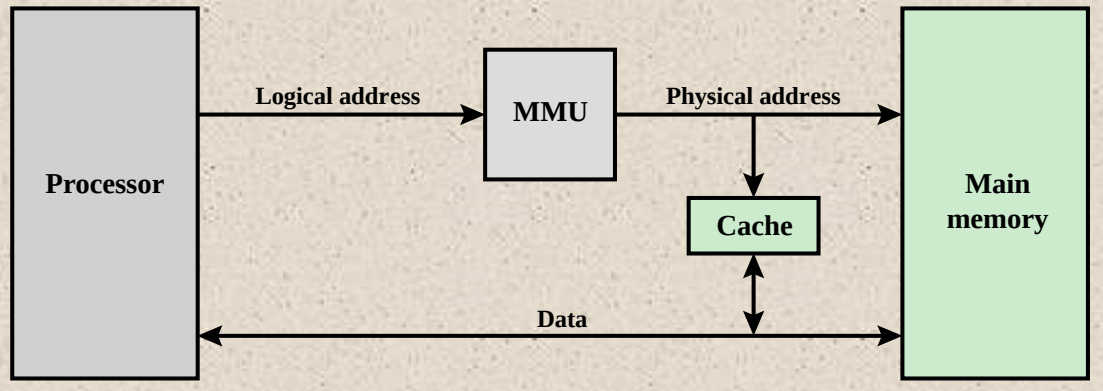
- Virtual memory
  - Programların fiziksel olarak mevcut ana bellek miktarına bakılmaksızın belleği mantıksal bir bakış açısıyla ele almasına izin veren özellik
  - Kullanıldığında, makine komutlarının adres alanları sanal adresler içerir
  - Ana bellekten okuma ve yazma işlemleri için, bir donanımsal bellek yönetim birimi (memory management unit-MMU) her sanal adresi ana bellekteki fiziksel bir adrese çevirir

Neredeyse tüm gömülü olmayan işlemciler ve birçok gömülü işlemci, sanal belleği (*virtual memory*) destekler.





(a) Logical Cache



(b) Physical Cache

Figure 4.7 Logical and Physical Caches

Sanal adresler kullanıldığında, sistem tasarımcısı önbelleği şekilde görüldüğü gibi işlemci ile MMU arasına veya MMU ile ana bellek arasına yerleştirmeyi seçebilir

Sanal önbellek (**virtual cache**) olarak da bilinen mantıksal önbellek (**logical cache**), verileri sanal adresleri kullanarak depolar. İşlemci, MMU'dan geçmeden önbelleğe doğrudan erişir. Fiziksel bir önbellek ise, ana bellek fiziksel adreslerini (**physical addresses**) kullanarak verileri depolar.

Mantıksal önbelleğin bariz bir **avantajı**, önbellek erişim hızının fiziksel bir önbellekten daha hızlı olmasıdır, çünkü önbellek MMU bir adres çevirisi gerçekleştirmeden önce yanıt verebilir.

Dezavantaj, çoğu sanal bellek sisteminin her uygulamaya aynı sanal bellek adres alanını sağlaması gerçeğiyle ilgilidir. Yani, her uygulama 0 adresinde başlayan bir sanal bellek görür. Bu nedenle, iki farklı uygulamadaki aynı sanal adres, iki farklı fiziksel adresi ifade eder. Bu nedenle, önbellek her bir «application context switch»te tamamen temizlenmeli veya bu adresin hangi sanal adres alanını ifade ettiğini belirlemek için önbelleğin her satırına fazladan bit eklenmelidir.

Processor	Type	Year of Introduction	L1 Cache[a]	L2 cache	L3 Cache
IBM 360/85	Mainframe	1968	16 to 32 kB	—	—
PDP-11/70	Minicomputer	1975	1 kB	—	—
VAX 11/780	Minicomputer	1978	16 kB	—	—
IBM 3033	Mainframe	1978	64 kB	—	—
IBM 3090	Mainframe	1985	128 to 256 kB	—	—
Intel 80486	PC	1989	8 kB	—	—
Pentium	PC	1993	8 kB/8 kB	256 to 512 KB	—
PowerPC 601	PC	1993	32 kB	—	—
PowerPC 620	PC	1996	32 kB/32 kB	—	—
PowerPC G4	PC/server	1999	32 kB/32 kB	256 KB to 1 MB	2 MB
IBM S/390 G6	Mainframe	1999	256 kB	8 MB	—
Pentium 4	PC/server	2000	8 kB/8 kB	256 KB	—
IBM SP	High-end server/ supercomputer	2000	64 kB/32 kB	8 MB	—
CRAY MTAb	Supercomputer	2000	8 kB	2 MB	—
Itanium	PC/server	2001	16 kB/16 kB	96 KB	4 MB
Itanium 2	PC/server	2002	32 kB	256 KB	6 MB
IBM POWER5	High-end server	2003	64 kB	1.9 MB	36 MB
CRAY XD-1	Supercomputer	2004	64 kB/64 kB	1MB	—
IBM POWER6	PC/server	2007	64 kB/64 kB	4 MB	32 MB
IBM z10	Mainframe	2008	64 kB/128 kB	3 MB	24-48 MB
Intel Core i7 EE 990	Workstaton/ server	2011	6 ´ 32 kB/32 kB	1.5 MB	12 MB
IBM zEnterprise 196	Mainframe/ Server	2011	24 ´ 64 kB/ 128 kB	24 ´ 1.5 MB	24 MB L3 192 MB L4

Table 4.3

Cache Sizes of Some Processors

[a] Two values separated by a slash refer to instruction and data caches.

[b] Both caches are instruction only; no data caches.

(Table can be found on page 134 in the textbook.)

Mapping Function

Mapping fonksionun seçimi, önbelleğin nasıl organize edileceğini belirler.

- Ana bellek bloklarından daha az önbellek satırı olduğundan, ana bellek bloklarını önbellek satırlarına eşlemek için bir algoritma gereklidir.

Ayrıca, hangi ana bellek bloğunun o anda bir önbellek hattını işgal ettiğini belirlemek için bir araca ihtiyaç vardır.

- Üç teknik kullanılabilir:

Direct

- En basit teknik
- Her ana bellek bloğunu yalnızca bir olası önbellek satırına (line) eşler

Associative

- Her ana bellek bloğunun önbelleğin herhangi bir satırına yüklenmesine izin verir
- Önbellek kontrol lojiği (cache control logic), bir bellek adresini yalnızca bir Tag ve bir Word alanı olarak yorumlar
- Bir bloğun önbellekte olup olmadığını belirlemek için, önbellek kontrol mantığı bir eşleşme için her satırın Etiketini aynı anda incelemelidir.

Set Associative

- Hem doğrudan hem de çağrışımlı yaklaşımların güçlü yönlerini sergileyen ve onların dezavantajlarını azaltan bir denge modeli

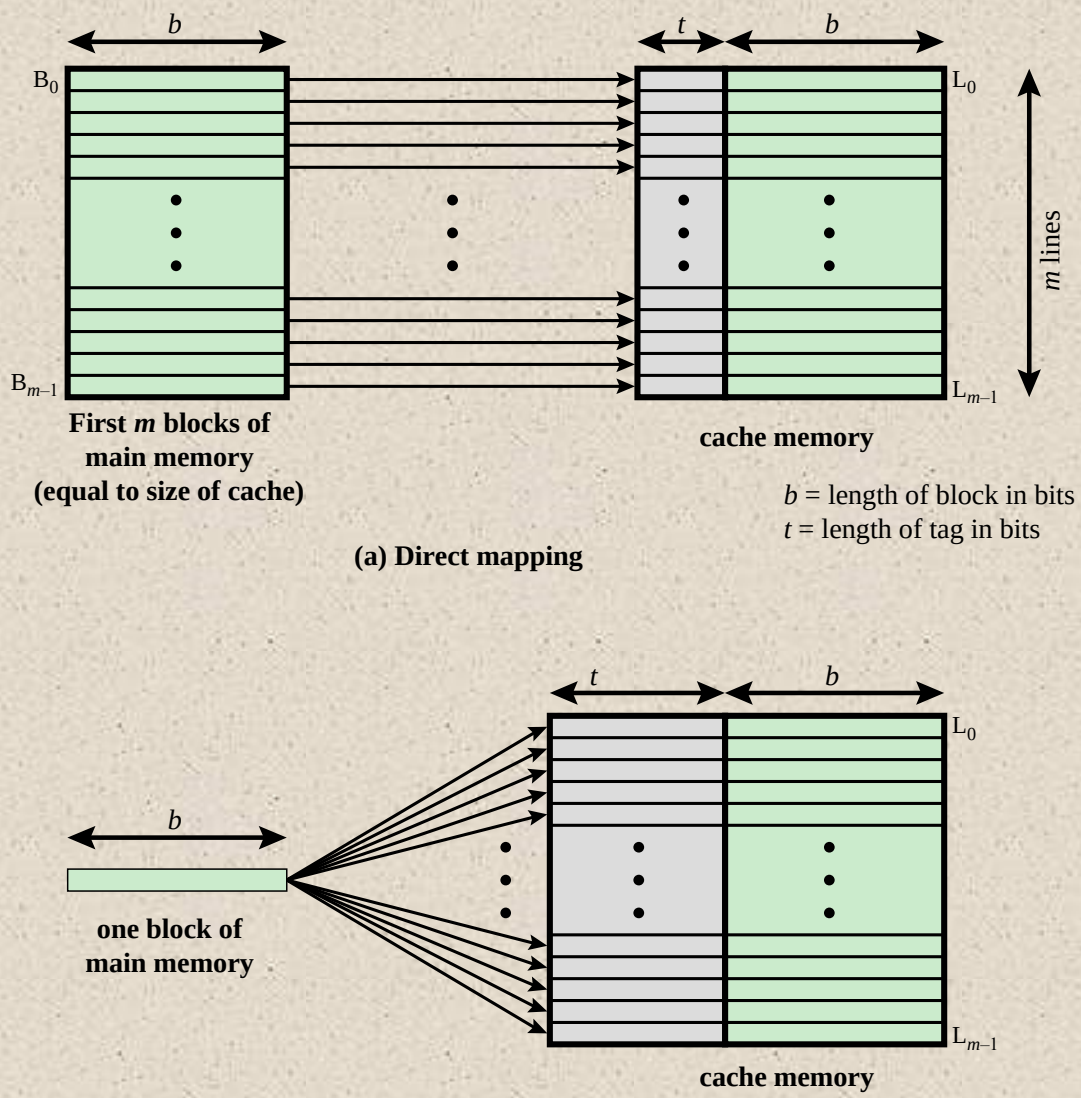


Figure 4.8 Mapping From Main Memory to Cache: Direct and Associative

The mapping is expressed as
i = j modulo m
where
i = cache line number
j = main memory block number
m = number of lines in the cache

Şekil 4.8a, ana belleğin ilk m bloğu için eşlemeyi gösterir.

Her ana bellek bloğu, önbelleğin benzersiz bir satırına eşlenir. Ana belleğin sonraki m bloğu aynı şekilde önbelleğe eşlenir; yani ana belleğin B_m bloğu önbelleğin L_0 satırına eşlenir, B_{m+1} bloğu L_1'e eşlenir, vesaire…

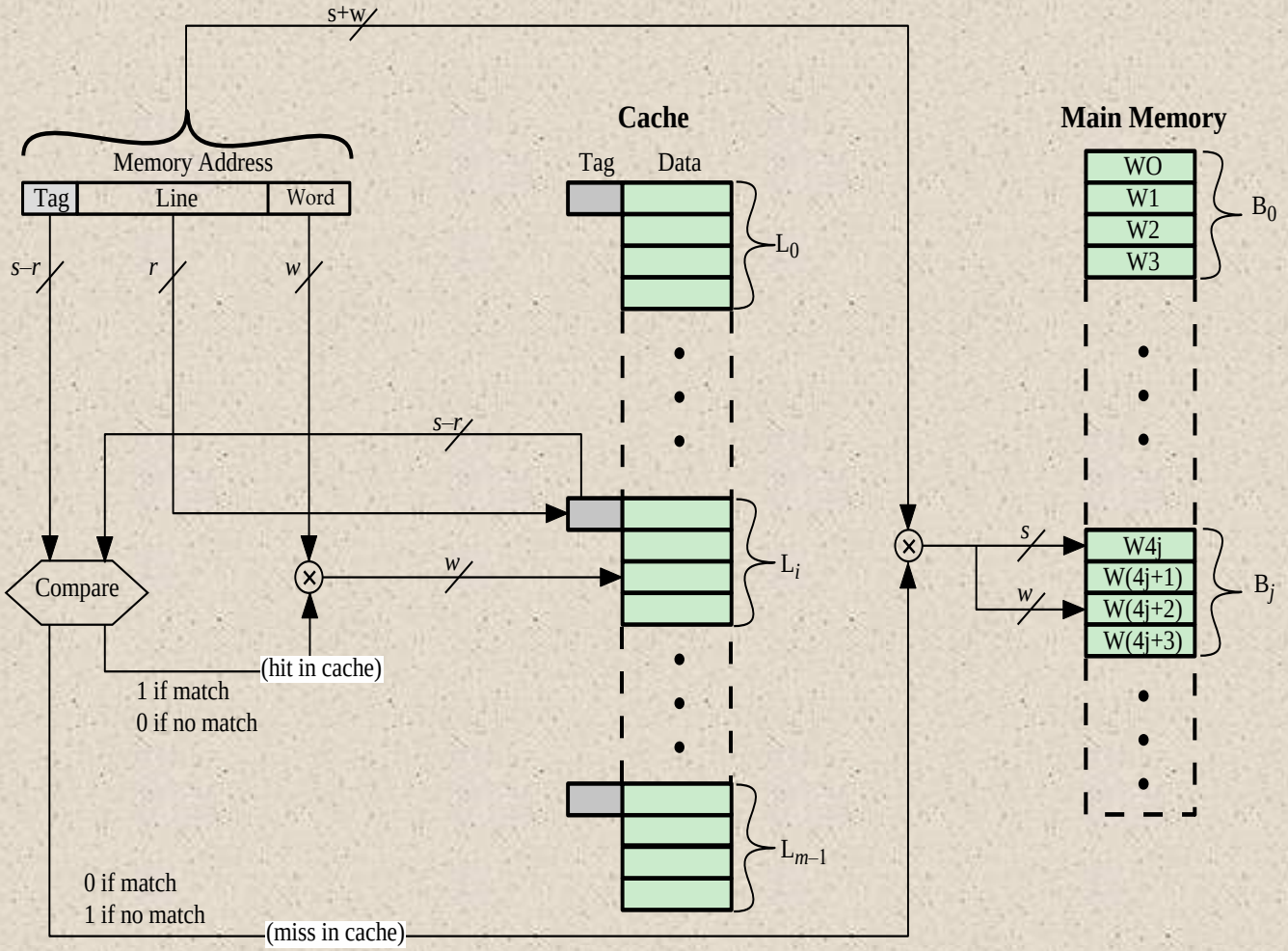


Figure 4.9 Direct-Mapping Cache Organization

Önbellek erişimi amacıyla, her ana bellek adresi üç alandan oluşuyor olarak görülebilir.

- En düşük değerli *w* bitleri, ana belleğin bir bloğu içindeki benzersiz bir word'u veya baytı tanımlar; modern makinelerin çoğunda adres bayt seviyesindedir.
- Kalan bitler ana belleğin 2^s bloğundan birini belirtir.

Önbellek lojiği, bu s adet bitlerin *s - r* bitlerini bir etiketi (*tag*) (en değerli kısım) ve *r* bitlerinden oluşan bir satır (*line*) alanı olarak yorumlar.

- Bu ikinci alan önbelleğin $m = 2^r$ satırlarından birini tanımlar.

- Address length = $(s + w)$ bits
- Number of addressable units = 2^{s+w} words or bytes
- Block size = line size = 2^w words or bytes
- Number of blocks in main memory = $\frac{2^{s+w}}{2^w} = 2^s$
- Number of lines in cache = $m = 2^r$
- Size of cache = 2^{r+w} words or bytes
- Size of tag = $(s - r)$ bits

Figure 4.10 shows our example system using direct mapping. In the example,
$m = 16K = 2^{14}$ and $i = j$ modulo 2^{14}. The mapping becomes

Cache Line	Starting Memory Address of Block
0	000000, 010000, …, FF0000
1	000004, 010004, …, FF0004
.	.
.	.
.	.
$2^{14} - 1$	00FFFC, 01FFFC, …, FFFFFC

Aynı satır numarasıyla eşleşen iki bloğun aynı etiket numarasına sahip olmadığına dikkat edin. Bu nedenle, başlangıç adresleri 000000, 010000,…, FF0000 olan bloklar sırasıyla 00, 01,…, FF etiket numaralarına sahiptir.

Bir okuma işlemi aşağıdaki gibi çalışır.
Önbellek sistemine 24 bitlik bir adres sunulur.

14 bitlik satır numarası, belirli bir satıra erişmek için önbelleğe bir dizin (*index*) olarak kullanılır.

8 bitlik etiket numarası o satırda o anda depolanan etiket numarasıyla eşleşirse, 2 bitlik kelime (word) numarası o satırdaki 4 bayttan birini seçmek için kullanılır.

Aksi takdirde, 22 bitlik etiket + satır alanı, ana bellekten bir bloğu almak için kullanılır.
Getirme için kullanılan gerçek adres, iki 0 bit ile birleştirilen 22 bitlik etiket + satırdır, böylece bir blok sınırından başlayarak 4 bayt getirilir.

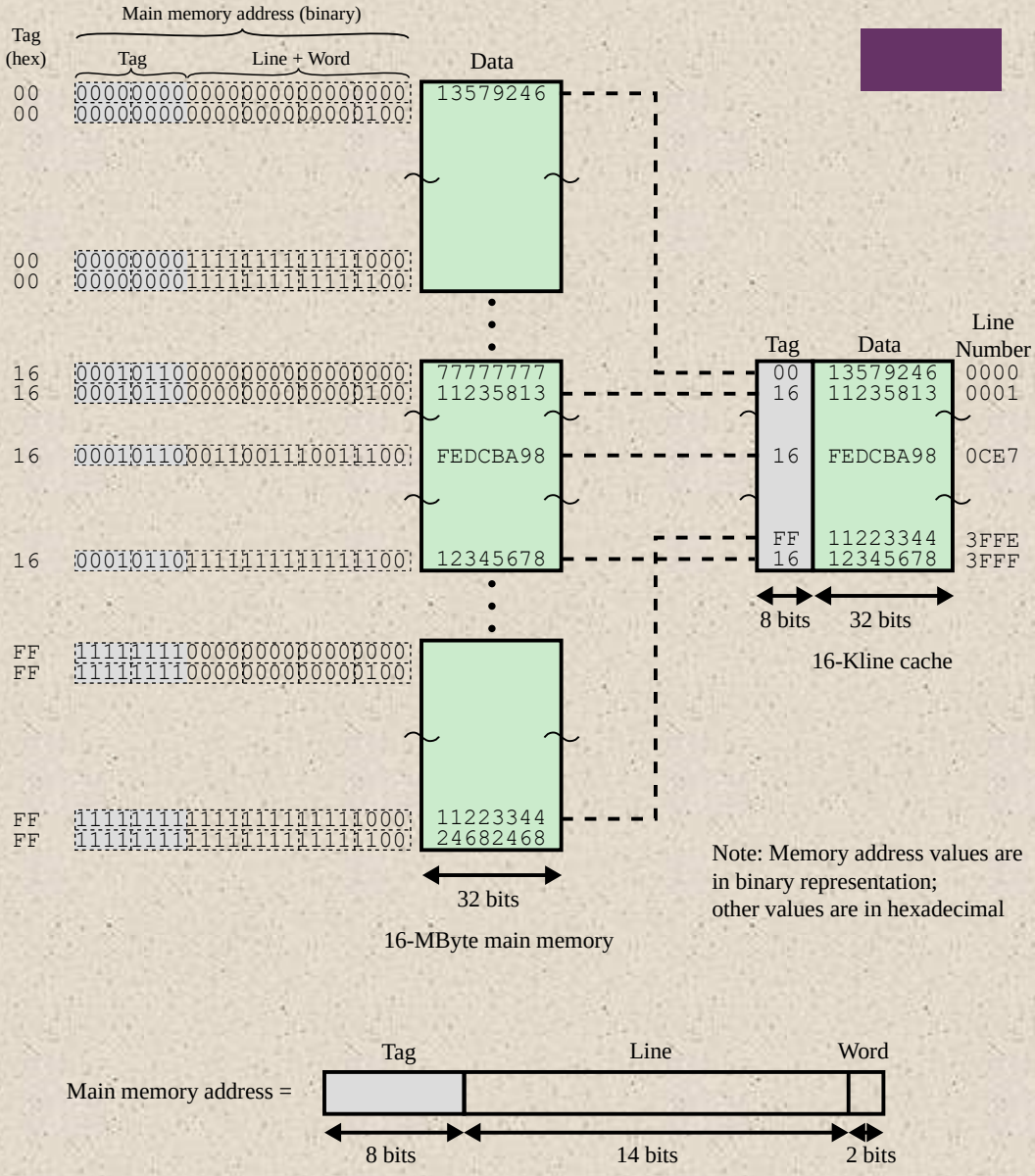
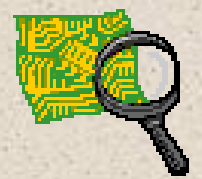


Figure 4.10 Direct Mapping Example

Direct Mapping Summary

- Address length = (s + w) bits
- Number of addressable units = 2^{s+w} words or bytes
- Block size = line size = 2^{w} words or bytes
- Number of blocks in main memory = $2^{s+w}/2^{w} = 2^{s}$
- Number of lines in cache = m = 2^{r}
- Size of tag = (s – r) bits

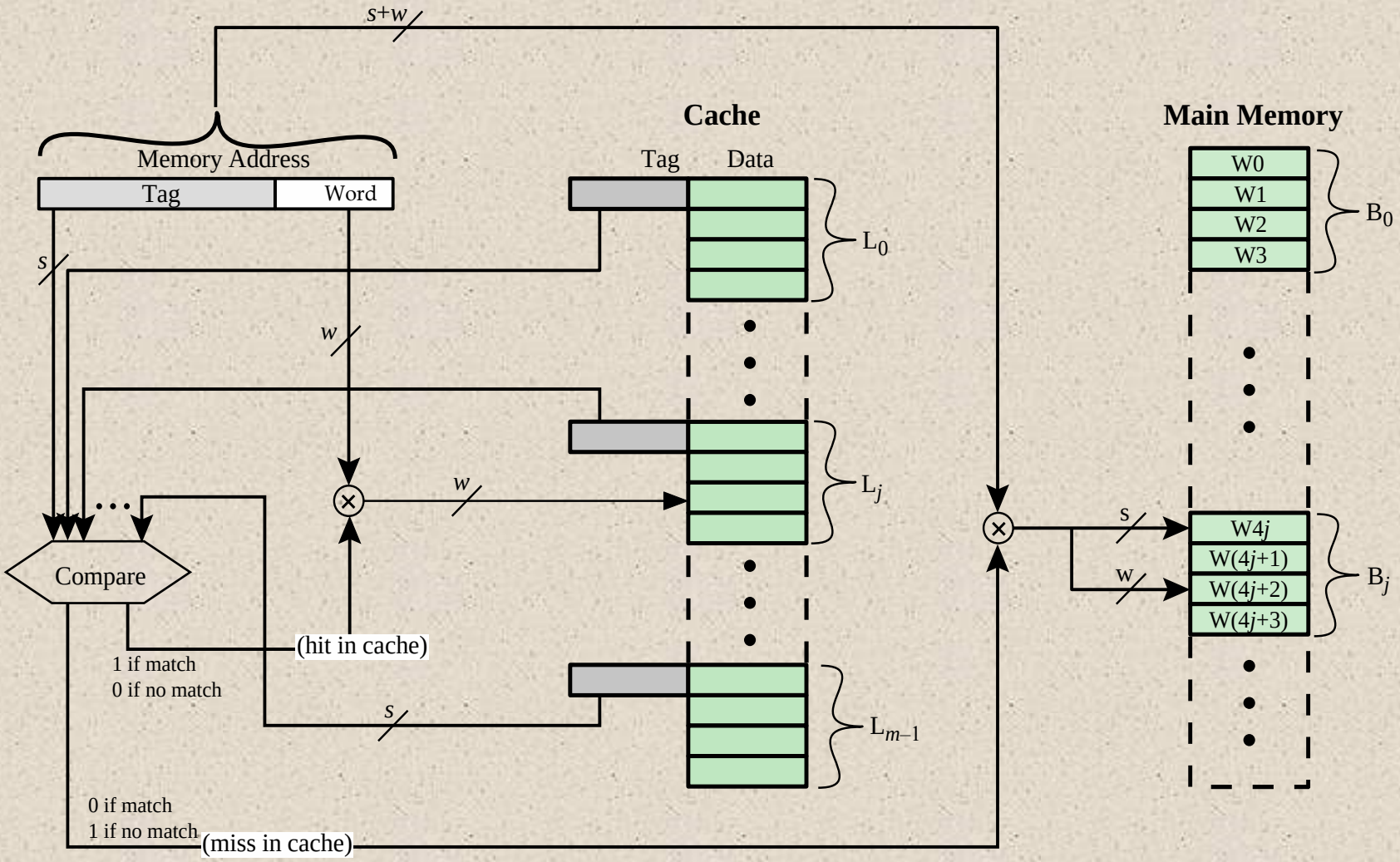


Doğrudan haritalama tekniğinin uygulanması basit ve ucuzdur. **Ana dezavantajı**, herhangi bir blok için sabit bir önbellek konumu olmasıdır. Bu nedenle, bir program aynı satıra eşlenen iki farklı bloktan tekrar tekrar kelimelere başvurursa, bloklar sürekli olarak önbellekte değiştirilecek (**swap**) ve isabet oranı düşük olacaktır (***thrashing*** olarak bilinen bir durum).

Victim Cache

- İsabetsizlik/Iskalama cezasını (***miss penalty***) düşürmenin bir yolu, tekrar gerekli olması durumunu gözeterek neyin atıldığını hatırlamaktır. Silinen veri zaten getirilmiş olduğundan, tekrar küçük bir maliyetle kullanılabilir. Bu tür bir geri dönüşüm, **victim cache** kullanılarak mümkündür
- Victim cache, başlangıçta, hızlı erişim süresini etkilemeden doğrudan eşlemeli (*direct mapping*) önbelleklerdeki çatışma ıskalamalarını (*conflict misses*) azaltmak için bir yaklaşım olarak önerildi
- Fully associative cache
- Tipik boyut 4 ila 16 önbellek satırıdır
- Doğrudan eşlemeli L1 önbelleği ile bir sonraki bellek düzeyi arasında bulunur

Associative mapping, her ana bellek bloğunun önbelleğin herhangi bir satırına yüklenmesine izin vererek direct mapping'in dezavantajının üstesinden gelir (Şekil 4.8b)



Bu durumda, önbellek kontrol lojiği bir bellek adresini sadece bir Etiket (*Tag*) ve bir Kelime (*Word*) alanı olarak yorumlar. Etiket alanı, bir ana bellek bloğunu benzersiz şekilde tanımlar. Bir bloğun önbellekte olup olmadığını belirlemek için, önbellek kontrol lojiği, eşleşme için her satırın etiketini aynı anda incelemelidir.

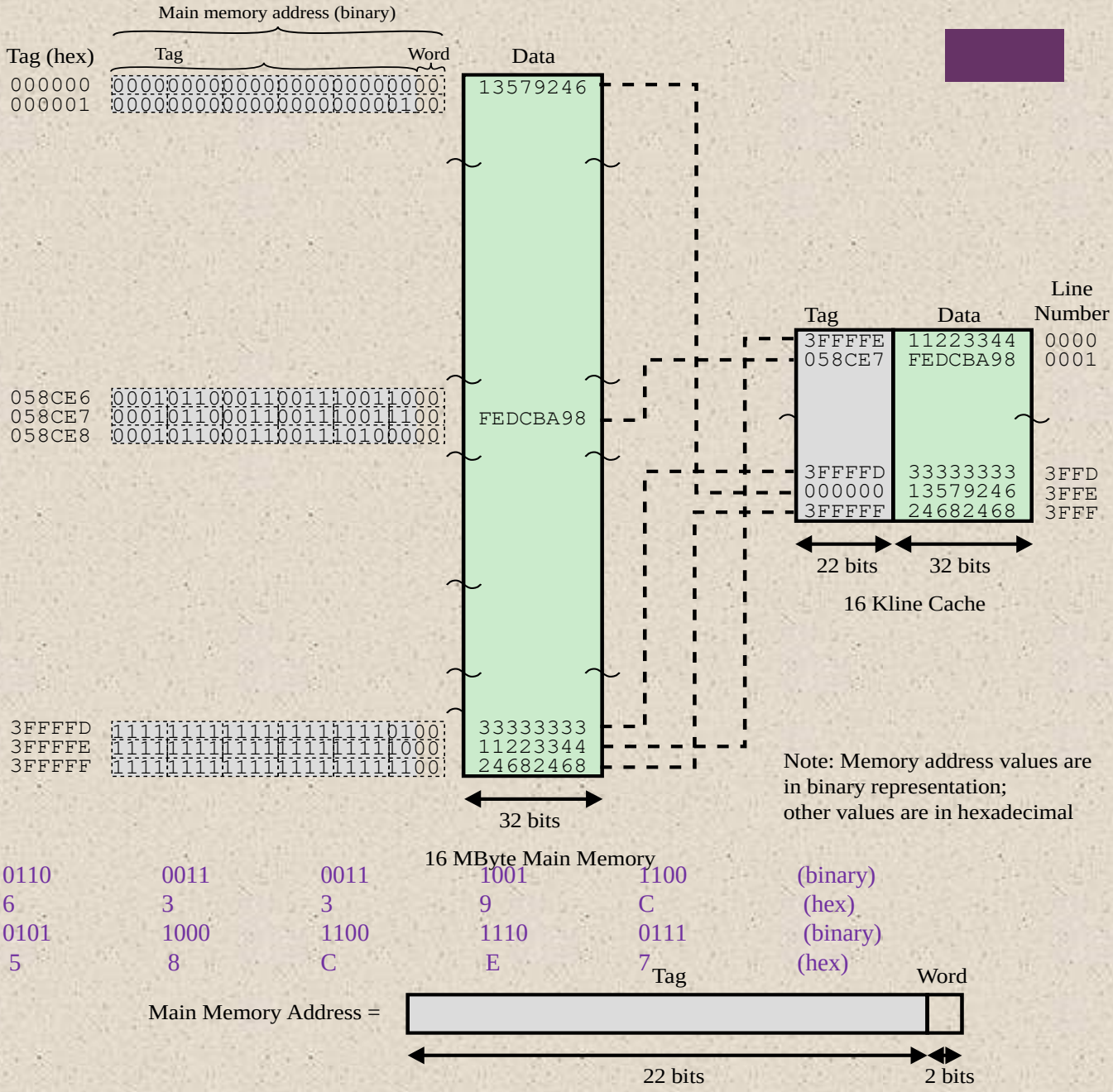
Figure 4.11 Fully Associative Cache Organization

Şekil 4.12, associative mapping kullanan bir örneği göstermektedir. Bir ana bellek adresi 22 bitlik bir etiket (Tag) ve 2 bitlik bir bayt numarasından oluşur.

22 bitlik etiket, önbellekteki her satır için 32 bitlik veri bloğu ile depolanmalıdır.

Etiketi oluşturan, adresin en soldaki (en değerli) 22 biti olduğuna dikkat edin.

Bu nedenle, 24 bitlik hexadecimal adres **16339C**, 22 bitlik etikete **058CE7** sahiptir. Bu, ikili gösterimde kolayca görülebilir:

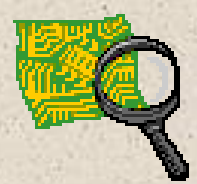


memory address	0001	0110	0011	0011	1001	1100	(binary)
	1	6	3	3	9	C	(hex)
tag (leftmost 22 bits)	00	0101	1000	1100	1110	0111	(binary)
	0	5	8	C	E	7	(hex)

Figure 4.12 Associative Mapping Example

Associative Mapping Summary

- Address length = (s + w) bits
- Number of addressable units = 2^{s+w} words or bytes
- Block size = line size = 2^{w} words or bytes
- Number of blocks in main memory = $2^{s+w}/2^{w} = 2^{s}$
- Size of tag = s bits



Associative mapping ile, önbelleğe yeni bir blok okunduğunda (getirildiğinde) hangi bloğun değiştirileceği konusunda esneklik vardır.
Bu bölümde daha sonra tartışılan değiştirme (*replacement*) algoritmaları, isabet oranını en üst düzeye çıkarmak için tasarlanmıştır.
Associative mapping **temel dezavantajı**, tüm önbellek hatlarının etiketlerini paralel olarak incelemek için gereken karmaşık devredir.

Set Associative Mapping

- Hem doğrudan hem de çağrışımlı yaklaşımların dezavantajlarını azaltırken güçlü yönlerini sergileyen yaklaşım
- Önbellek birkaç setten oluşur
- Her set bir dizi satır (*lines*) içerir
- Belirli bir blok, belirli bir kümedeki herhangi bir satıra eşlenir
- Örneğin set başına 2 satır
  - 2 way associative mapping
  - Belirli bir blok, yalnızca bir setteki 2 satırdan birinde olabilir

Set Associative Mapping

- Bu durumda, önbellek, her biri bir dizi satırdan oluşan bir dizi setten oluşur. İlişkiler:

$m = v * k$

$i = j \text{ modulo } v$

where

i = cache set number

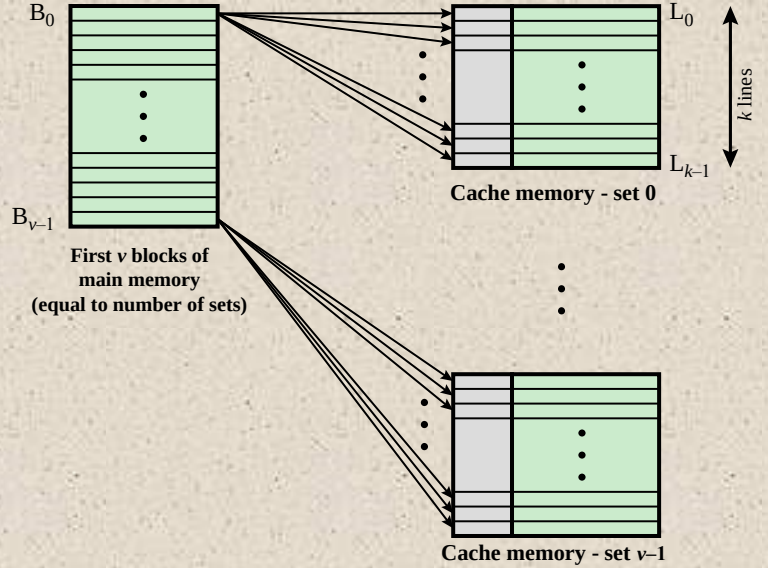
j = main memory block number

m = number of lines in the cache

v = number of sets

k = number of lines in each set

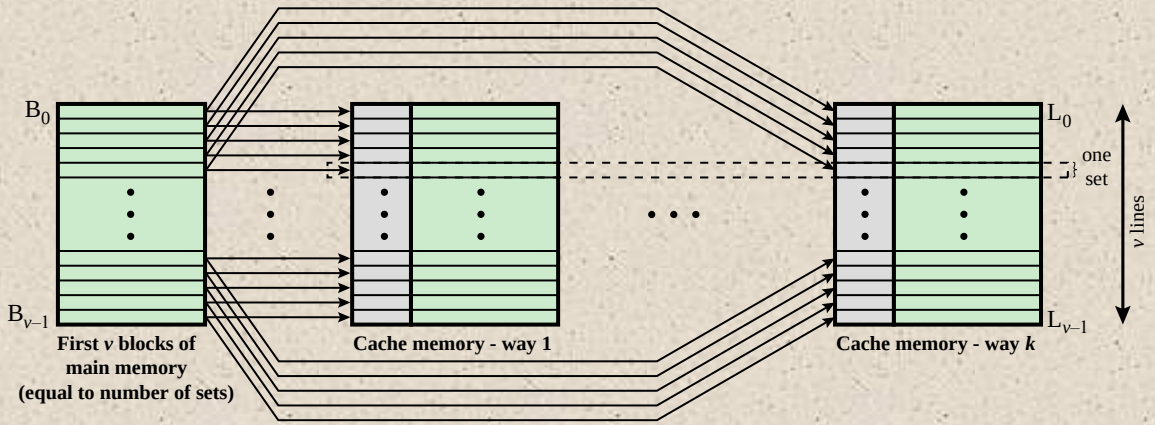
Bu, *k-way set-associative mapping* olarak adlandırılır. Set-associative mapping ile, blok $\mathbf{B_j}$, **j** kümesinin herhangi bir satırına eşlenebilir.



(a) *v* associative-mapped caches

Şekil 4.13a, ana belleğin ilk v blokları için bu eşlemeyi gösterir. *associative mapping*'de olduğu gibi, her kelime birden çok önbellek satırına eşlenir.
Set-associative mapping için, her kelime belirli bir setteki tüm önbellek satırlarına eşlenir, böylece ana bellek bloğu B0, set 0'a eşlenir ve bu böyle devam eder.

Bu nedenle, set-associative cache fiziksel olarak n «*associative cahches*» olarak uygulanabilir.



(b) *k* direct-mapped caches

Şekil 4.13b'de gösterildiği gibi, set-associative cache, «*k direct mapping caches*» olarak uygulamak da mümkündür.

Her doğrudan eşlemeli önbellek, *v* satırlarından oluşan bir yol (*way*) olarak adlandırılır.

Ana belleğin ilk v satırları, her yolun *v* satırlarına doğrudan eşlenir; ana belleğin sonraki v satırı grubu benzer şekilde eşlenir ve bu böyle devam eder.

Figure 4.13 Mapping From Main Memory to Cache: *k*-way Set Associative

The direct-mapped implementation is typically used for **small degrees of associativity** (small values of k) while the associative-mapped implementation is typically used for higher degrees of associativity

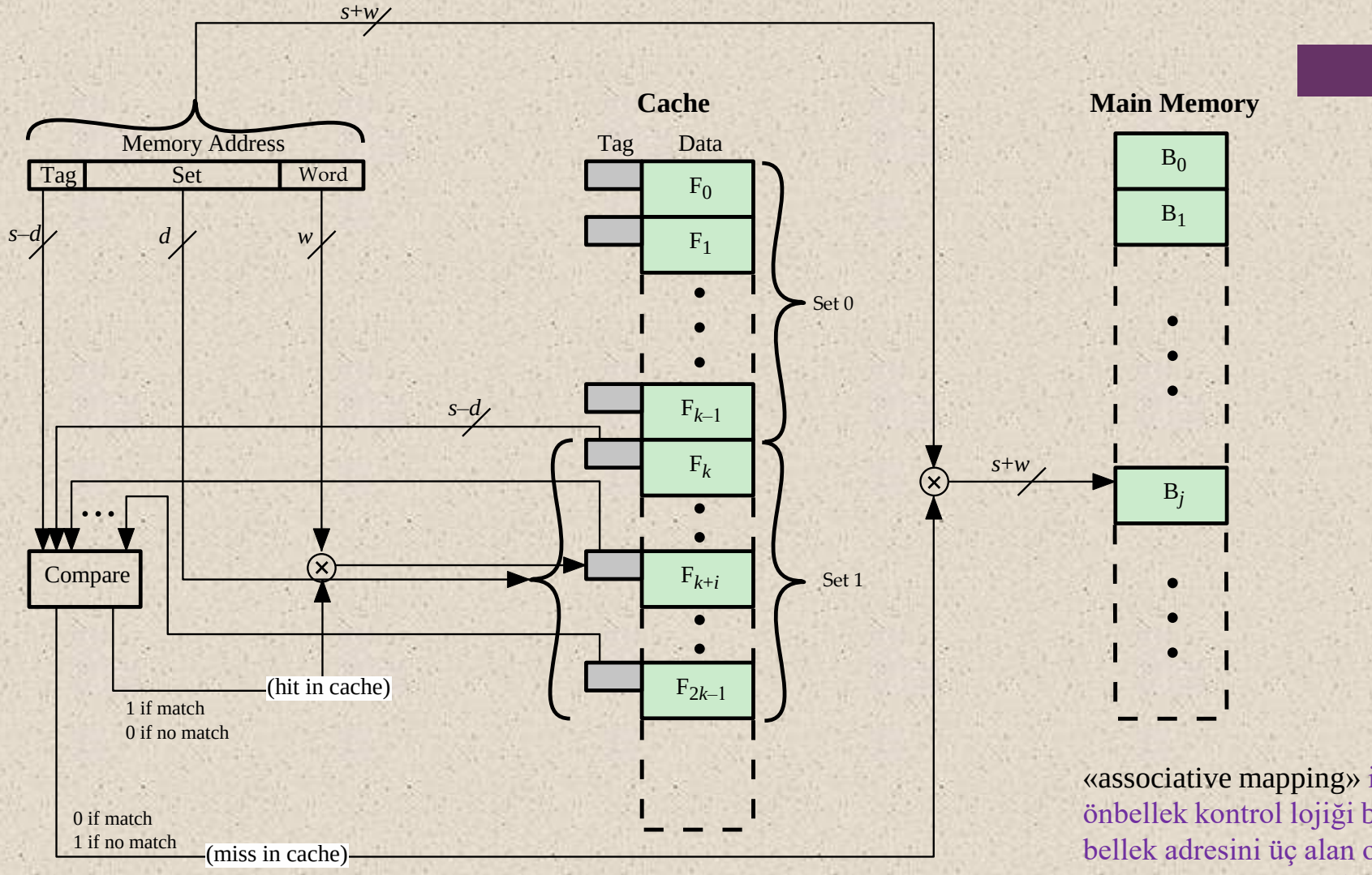


Figure 4.14 *k*-Way Set Associative Cache Organization

«*fully associative mapping*»'de bir bellek adresindeki etiket oldukça büyüktür ve önbellekteki her satırın etiketi ile karşılaştırılmalıdır. «*k-way set-associative mapping*»'*de ise*, bir bellek adresindeki etiket çok daha küçüktür ve yalnızca tek bir set içindeki *k* adet etiket ile karşılaştırılır.

«associative mapping» için, önbellek kontrol lojiği bir bellek adresini üç alan olarak yorumlar: Etiket (Tag), Küme (Set) ve Word.
d set bitleri, v = 2^d setlerden birini belirtir.
Etiket ve Set alanlarının s biti ana belleğin 2^s bloğundan birini belirtir.

Set Associative Mapping Summary

- Address length = (s + w) bits
- Number of addressable units = 2^{s+w} words or bytes
- Block size = line size = 2^{w} words or bytes
- Number of blocks in main memory = $2^{s+w}/2^{w}=2^{s}$
- Number of lines in set = k
- Number of sets = v = 2^{d}
- Number of lines in cache = m=kv = k * 2^{d}
- Size of cache = $k * 2^{d+w}$ words or bytes
- Size of tag = (s – d) bits

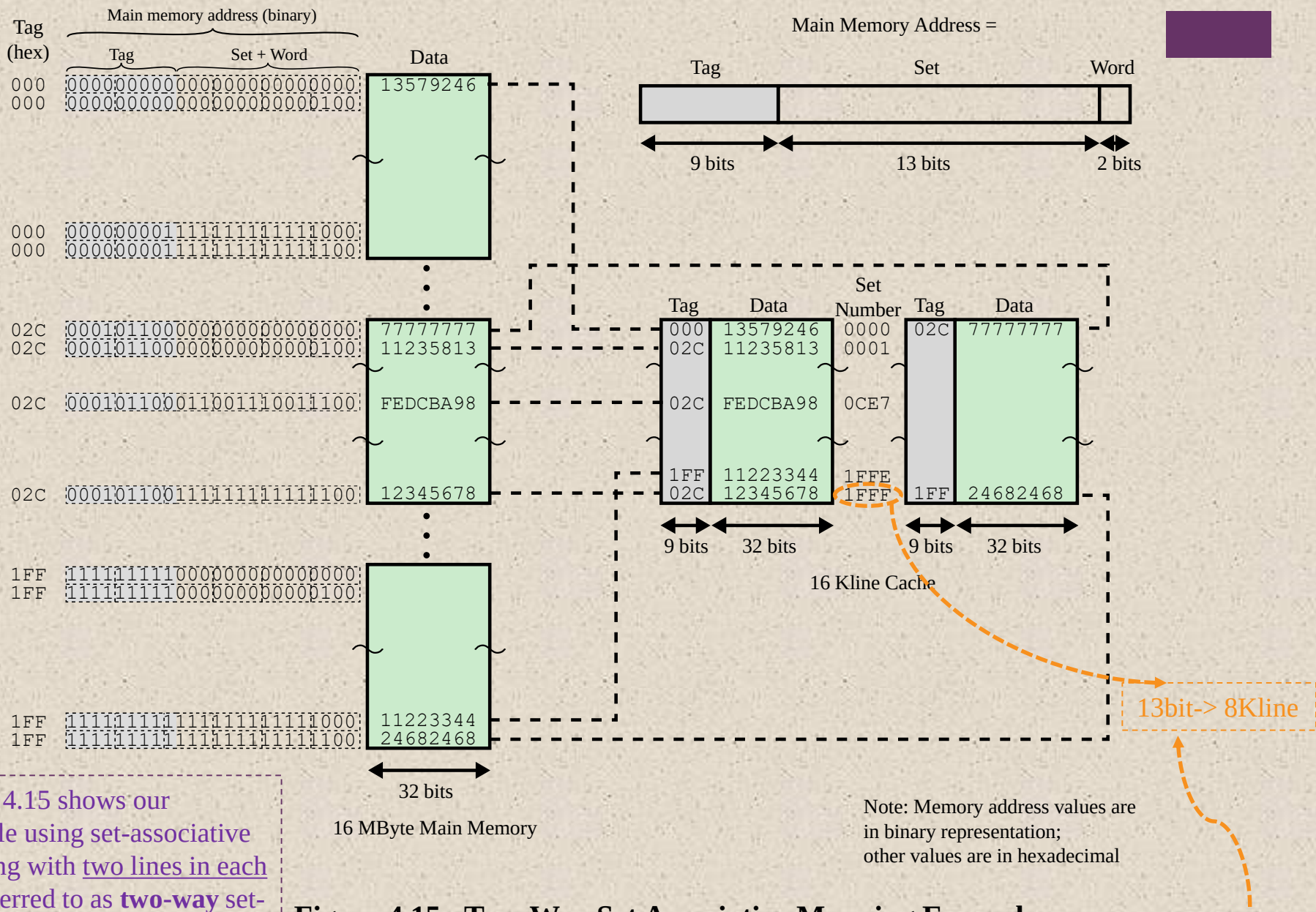


Figure 4.15 shows our example using set-associative mapping with two lines in each set, referred to as **two-way** set-associative.

Figure 4.15 Two-Way Set Associative Mapping Example

The 13-bit set number identifies a unique set of two lines within the cache. It also gives the number of the block in main memory, modulo 2^{13}

- 13 bitlik set numarası **(13-bit set number)**, önbellekteki benzersiz bir 2-satır setini (**a unique set of two lines**) tanımlar. Aynı zamanda ana bellekteki bloğun numarası olan modulo 2^{13} 'ü verir. Bu, blokların satırlara eşlenmesini belirler.

- Bu nedenle, ana bellek haritasının 000000, 008000,…, FF8000 blokları önbellek seti 0'a yerleştirilir. Bu bloklardan herhangi biri setteki iki satırdan birine (either of the two lines in the set) yüklenebilir.

- Aynı önbellek setiyle eşleşen iki bloğun aynı etiket numarasına sahip olmadığını unutmayın. Bir okuma işlemi için, 13 bitlik set numarası, hangi iki satırlık setin inceleneceğini belirlemek için kullanılır.

- Setteki her iki satır da erişilecek adresin etiket numarası ile eşleşme (*match*) açısından incelenir.

Set Associative Example

- "***two-way set-associative***" önbellek 16 baytlık satırlara (*block size*) ve toplam 8 kbayt boyuta sahiptir. 64-Mbyte ana bellek, bayt adreslenebilirdir (*byte addressable*). Buna göre ana bellek adreslerinin formatını gösterin.

Çözüm

Önbellkteki satır sayısı= 8Kb/16 byte=512

Dolayısıyla 2 satırlı 256 adet set vardır

- ve bu setleri belirtmek için **8 bit'e** ihtiyaç vardır.

64-MB lık ana bellek için

- 26-bitlik adres gerekir.

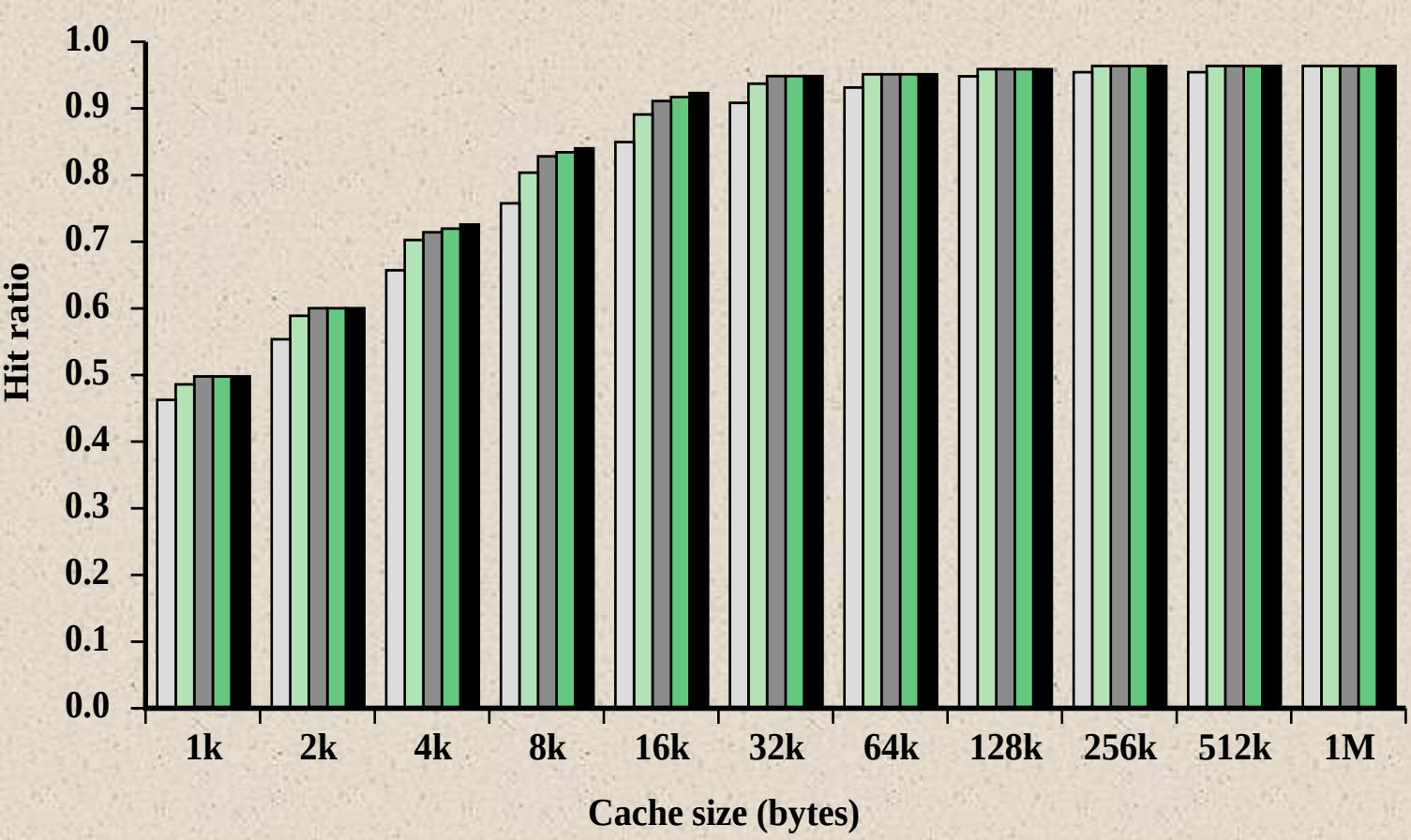
Ana bellek 64-Mb/16 byte =2^{22} bloktan oluşur

- Dolayısıyla (set+tag) 22 bit olmak zorundadır.

Bu sebeple tag uzunluğu 14 bittir ve word alanı uzunluğu da 4 bittir.

Ana bellek adresi =

TAG	SET	WORD
14	8	4



«direct» ve «two-way set associative» arasındaki performans farkı, en az 64 kB önbellek boyutuna kadar anlamlıdır.

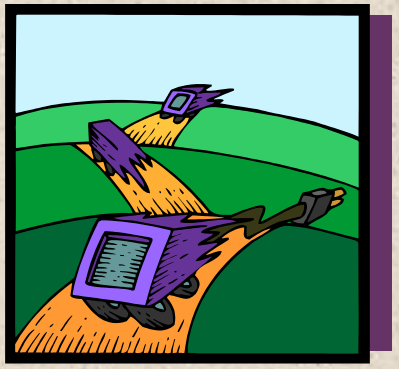
Ayrıca, 4 kB'de «two-way» ve «four-way» arasındaki farkın, önbellek boyutunda 4 kB'den 8 kB'ye geçişteki farktan çok daha az olduğu görülmektedir.

- Önbelleğin karmaşıklığı, «associativity» ile birlikte orantılı olarak artar ve bu durumda, önbellek boyutunu 8 veya hatta 16 Kbayta çıkarmaya karşı haklı gösterilemez.
- Unutulmaması gereken son bir nokta da, yaklaşık 32 kB'nin ötesinde, önbellek boyutundaki artışın performansta önemli bir artış getirmemesidir.

Figure 4.16 Varying Associativity over Cache Size

Şekil 4.16, önbellek boyutunun bir fonksiyonu olarak set-associative cache performansının bir simülasyon çalışmasının sonuçlarını gösterir. Şekil 4.16'nın sonuçları, bir GCC derleyicisinin yürütülmesini simüle etmeye dayanmaktadır. Farklı uygulamalar farklı sonuçlar verebilir.

Replacement Algorithms



- Önbellek doldurulduktan sonra, önbelleğe yeni bir blok getirildiğinde, mevcut bloklardan biriyle değiştirilmelidir.
- Doğrudan eşleme (*direct mapping*) için herhangi bir belirli blok için yalnızca bir olası satır vardır ve hiçbir seçim mümkün değildir
- *associative* ve *set-associative* teknikler için bir yer değiştirme algoritmasına ihtiyaç vardır.
- Yüksek hıza ulaşmak için, bir algoritma donanımsal olarak uygulanmalıdır

En yaygın değiştirme (*replacement*) algoritmaları şunlardır:

- Least recently used (LRU)
  - En etkili
  - Önbellekte referans verilmeden en uzun süredir bulunan bloğu değiştirin
  - Uygulama basitliği nedeniyle, LRU en popüler değiştirme
- First-in-first-out (FIFO)
  - Önbellekte en uzun süredir olan bloğu değiştirin
  - Round-robin veya dairesel buffer tekniği olarak kolayca uygulanır
- Least frequently used (LFU)
  - En az referansla karşılaşan bloğu değiştirin
  - Her satır ile bir sayaç ilişkilendirilerek uygulanabilir

«two-way set associative» için bu kolayca uygulanır. Her satır bir «USE» biti içerir. Bir satıra referans verildiğinde, USE biti 1'e ayarlanır ve bu kümedeki diğer satırın «USE» biti 0'a ayarlanır. Bir blok set'e okunacaksa/getirilecekse, «USE» biti 0 olan satır kullanılır. Son dönemde kullanılan bellek konumlarının referans alınma olasılığının daha yüksek olduğunu varsaydığımız için, LRU en iyi isabet oranını vermelidir.

LRU'nun «fully associative cache» için uygulanması da nispeten kolaydır. Önbellek mekanizması, önbellekteki tüm satırlar için ayrı bir dizin listesi tutar. Bir satıra referans verildiğinde, listenin başına gider. Değiştirme için listenin en altındaki satır kullanılır.

Write Policy

Önbellekte bulunan bir bloğun değiştirilmesi gerektiğinde, dikkate alınması gereken iki durum vardır:

Önbellekteki eski blok modifiye edilmediyse, bu eski bloğu ana belleğe yazmadan yeni bir blokla üzerine yazılabilir (*overwritten*).

Önbelleğin ilgili satırındaki bir kelime üzerinde en az bir yazma işlemi gerçekleştirilmişse, yeni bloğu getirmeden önce önbellek satırını bellek bloğuna yazarak ana bellek güncellenmelidir.

Üstesinden gelinmesi gereken iki sorun var:

Birden fazla cihazın ana belleğe erişimi olabilir

Aynı veri yoluna birden fazla işlemci eklendiğinde ve her işlemcinin kendi yerel önbelleği olduğunda daha karmaşık bir sorun ortaya çıkar - bir önbellekte bir kelime değiştirilirse, diğer önbelleklerdeki bir kelimeyi muhtemelen geçersiz kılabilir.

Write Through and Write Back

- **Write through**
  - En basit teknik
  - Tüm yazma işlemleri önbellek ile birlikte ana belleğe de yapılır
  - Bu tekniğin ana dezavantajı, önemli miktarda bellek trafiği oluşturması ve bir darboğaz (*bottleneck*) oluşturabilmesidir.
- **Write back**
  - Bellek yazmalarını en aza indirir
  - Güncellemeler yalnızca önbellekte yapılır
  - Ana belleğin ilgili bölümleri geçersizdir ve bu nedenle I/O modülleri tarafından erişime yalnızca önbellek yoluyla izin verilebilir
  - Bu, karmaşık devre ve potansiyel bir darboğaz oluşturur

Bir güncelleme (update) gerçekleştiğinde, satırla ilişkili bir «**dirty bit** » veya «**use bit**» set edilir. Daha sonra, bir blok değiştirildiğinde, ancak ve ancak dirty bit'i set edilmişse ana belleğe geri yazılır.

Deneyimler, yazılan bellek başvurularının yüzdesinin% 15 seviyesinde olduğunu göstermiştir. Bununla birlikte, HPC uygulamaları için, bu sayı% 33'e (vektör-vektör çarpımı) yaklaşabilir ve % 50'ye kadar çıkabilir (matris transpozu).

Line Size

Bir veri bloğu alındığında ve önbelleğe yerleştirildiğinde, yalnızca istenen kelime değil, aynı zamanda bazı bitişikteki kelimeler de alınır.

Blok boyutu arttıkça önbelleğe daha fazla yararlı veriler getirilir

İki özel etki devreye girer:

- Daha büyük bloklar, bir önbelleğe sığan blokların sayısını azaltır
- Bir blok büyüdükçe, her bir ek kelime istenen kelimeden daha uzaktır ve bu nedenle yakın gelecekte ihtiyaç duyulma olasılığı azalır.

Her blok getirme, eski önbellek içeriğinin üzerine yazdığından (*overwrite*), az sayıda blok, getirildikten kısa bir süre sonra verilerin üzerine yazılmasına neden olur.

Blok boyutu arttıkça, yerellik ilkesi (***principle of locality***) nedeniyle isabet oranı (principle of locality) ilk başta artacaktır.

Blok daha da büyüdükçe isabet oranı düşmeye başlayacak ve yeni getirilen bilgileri (*newly fetched information*) kullanma olasılığı, değiştirilmesi gereken (*to be replaced*) bilgileri yeniden kullanma olasılığından daha düşük hale gelecektir.

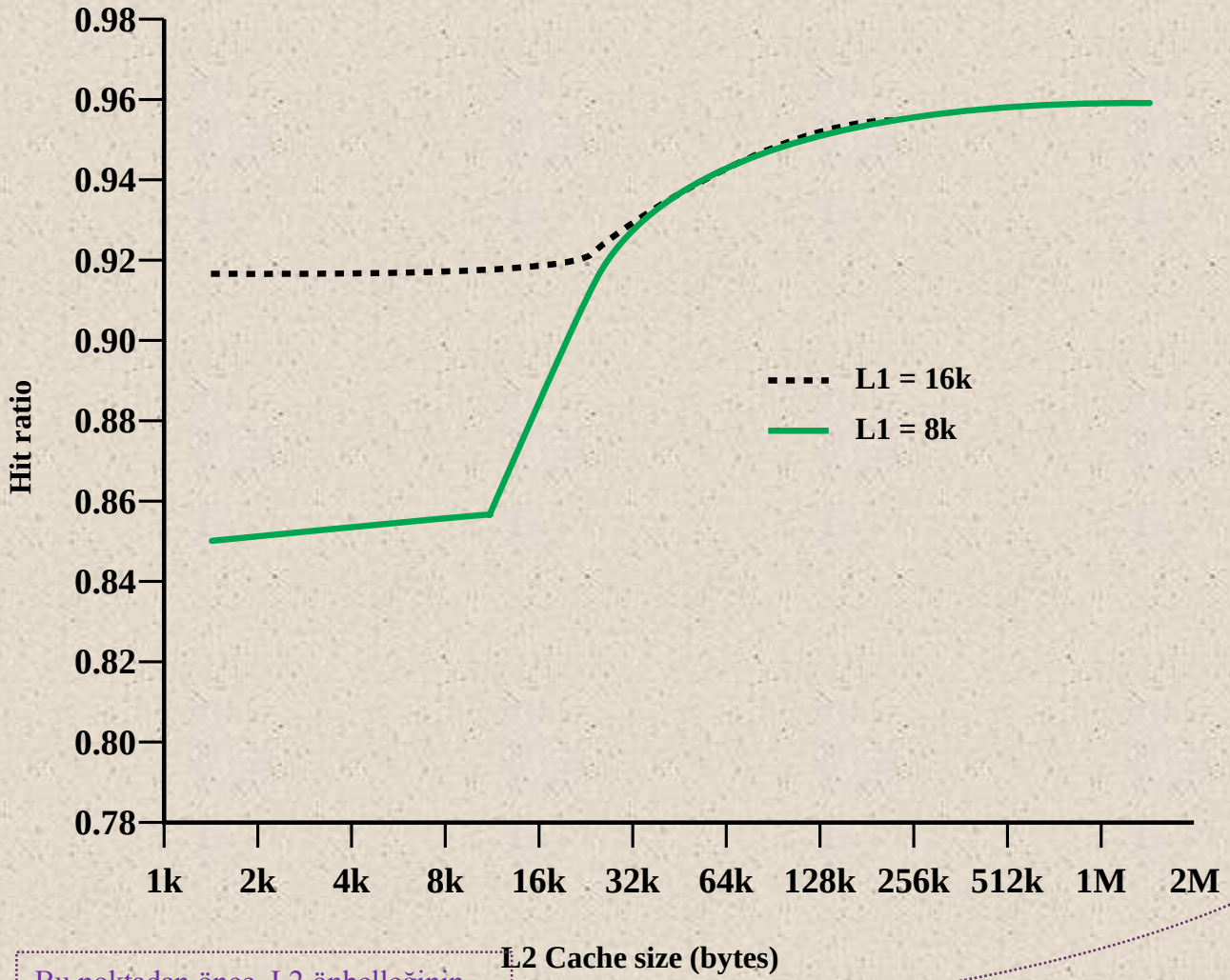
Blok boyutu (*block size*) ile isabet oranı (*hit ratio*) arasındaki ilişki, belirli bir programın yerellik özelliklerine bağlı olarak karmaşıktır ve kesin bir optimum değer bulunamamıştır. 8 ila 64 baytlık bir boyut, optimuma makul ölçüde yakın görünüyor. HPC sistemleri için en sık 64 ve 128 baytlık önbellek satır boyutları kullanılır.

Multilevel Caches

- Lojik yoğunluğu arttıkça, işlemci ile aynı yongada bir önbelleğe sahip olmak mümkün hale geldi
- Çipin içerisindeki önbellek (*on-chip cache*), işlemcinin harici veri yolu aktivitesini azaltır ve yürütme/icra süresini hızlandırır ve genel sistem performansını artırır
  - Talep edilen komut veya veriler yonga üzerindeki önbellekte bulunduğunda, veri yolu erişimi (*bus access*) ortadan kalkar
  - Çip içindeki önbellek erişimleri, sıfır bekleme durumlu veri yolu döngülerinden (***zero-wait state bus cycles***) bile önemli ölçüde daha hızlı tamamlanacaktır.
  - Bu süre zarfında veri yolu diğer transferleri desteklemek için serbesttir
- Two-level cache:
  - Internal cache designated as level 1 (L1)
  - External cache designated as level 2 (L2)

Çip içerisine önbelleğin dahil edilmesi, yonga dışı veya harici bir önbelleğin hala arzu edilir olup olmadığı sorusunu açık bırakır. Genellikle cevap evettir ve çoğu modern tasarım hem çip üzerinde hem de harici önbellekleri içerir. Bu türden en basit organizasyon, iki seviyeli önbellek olarak bilinir.

- Bir L2 önbellek kullanımından kaynaklanan potansiyel tasarruf, hem L1 hem de L2 önbelleklerindeki isabet oranlarına bağlıdır.
- Çok seviyeli önbelleklerin kullanımı, boyut (*size*), yer değiştirme (*replacement*) algoritması ve yazma ilkesi (*write policy*) dahil önbelleklerle ilgili tüm tasarım sorunlarını karmaşık hale getirir.



Bu noktadan önce, L2 önbelleğinin toplam önbellek performansı üzerinde çok az etkisi vardır.

- Şekil, her iki önbelleğin de aynı satır boyutuna sahip olduğunu varsayar ve toplam isabet oranını gösterir. Yani, istenen veriler L1 veya L2 önbelleğinde görünürse bir isabet sayılır.
- Şekil, L2'nin L1 boyutuna göre toplam isabetler üzerindeki etkisini göstermektedir.
  - ✓ L2, L1 önbellek boyutunun **en az iki katı** olana kadar toplam önbellek isabet sayısı üzerinde çok az etkiye sahiptir.
  - ✓ 8 Kbaytlık bir L1 önbelleği için eğimin en dik kısmının 16 Kbaytlık bir L2 önbelleği için olduğuna dikkat edin.
  - ✓ Yine 16 Kbaytlık bir L1 önbellek için, eğrinin en dik kısmı 32 Kbaytlık bir L2 önbellek boyutu içindir.

Figure 4.17 Total Hit Ratio (L1 and L2) for 8 Kbyte and 16 Kbyte L1

Şekil 4.17, önbellek boyutunun bir fonksiyonu olarak iki seviyeli önbellek (two-level cache) performansının bir simülasyon çalışmasının sonuçlarını göstermektedir.

Unified Versus Split Caches

Çip içerisinde/üzerinde önbellek ilk kez ortaya çıktığında, tasarımların çoğu hem verilere hem de komutlara referansları depolamak için kullanılan tek bir önbellekten oluşuyordu.

- (Sonraları) Önbelleği bölmek yaygın hale geldi:
  - Birisi komutlara (*instructions*) tahsis edilmiştir
  - Diğeri ise verilere (*data*)
  - Her ikisi de aynı seviyede, tipik olarak iki L1 önbelleği olarak bulunur
- Birleşik önbelleğin (*unified cache*) avantajları:
  - Daha yüksek isabet oranı (*Higher hit rate*)
    - Komut ve veri alımları yükünü otomatik olarak dengeler
    - Yalnızca bir önbelleğin tasarlanması ve uygulanması
- Trend, L1'de önbellekleri bölmek ve daha yüksek seviyeler için birleştirilmiş önbellekleri kullanmaktır
- Bölünmüş önbelleğin (*split cache*) avantajları:
  - Komut getirme / kod çözme birimi (*instruction fetch/decode unit*) ile yürütme birimi (*execution unit*) arasındaki önbellek çekişmesini ortadan kaldırır
    - *Pipelining* yapısında önemlidir

Problem	Solution	Processor on which Feature First Appears
External memory slower than the system bus. *Harici bellek sistem veri yolundan daha yavaş ise*	Add external cache using faster memory technology.	386
Increased processor speed results in external bus becoming a bottleneck for cache access. *Artan işlemci hızı, harici veri yolunun önbellek erişimi için bir darboğaz haline gelmesine neden olur.*	Move external cache on-chip, operating at the same speed as the processor.	486
Internal cache is rather small, due to limited space on chip *Çipteki sınırlı alan nedeniyle dahili önbellek oldukça küçüktür*	Add external L2 cache using faster technology than main memory	486
Contention occurs when both the Instruction Prefetcher and the Execution Unit simultaneously require access to the cache. In that case, the Prefetcher is stalled while the Execution Unit's data access takes place. *Hem Komut Önceden Getirici hem de Yürütme Birimi aynı anda önbelleğe erişim gerektirdiğinde ihtilaf oluşur*	Create separate data and instruction caches.	Pentium
Increased processor speed results in external bus becoming a bottleneck for L2 cache access. *Artan işlemci hızı, harici veri yolunun L2 önbellek erişimi için bir darboğaz haline gelmesine neden olur.*	Create separate back-side bus that runs at higher speed than the main (front-side) external bus. The BSB is dedicated to the L2 cache.	Pentium Pro
	Move L2 cache on to the processor chip.	Pentium II
Some applications deal with massive databases and must have rapid access to large amounts of data. The on-chip caches are too small.	Add external L3 cache.	Pentium III
	Move L3 cache on-chip.	Pentium 4

Table 4.4

Intel Cache Evolution

(Table is on page 150 in the textbook.)

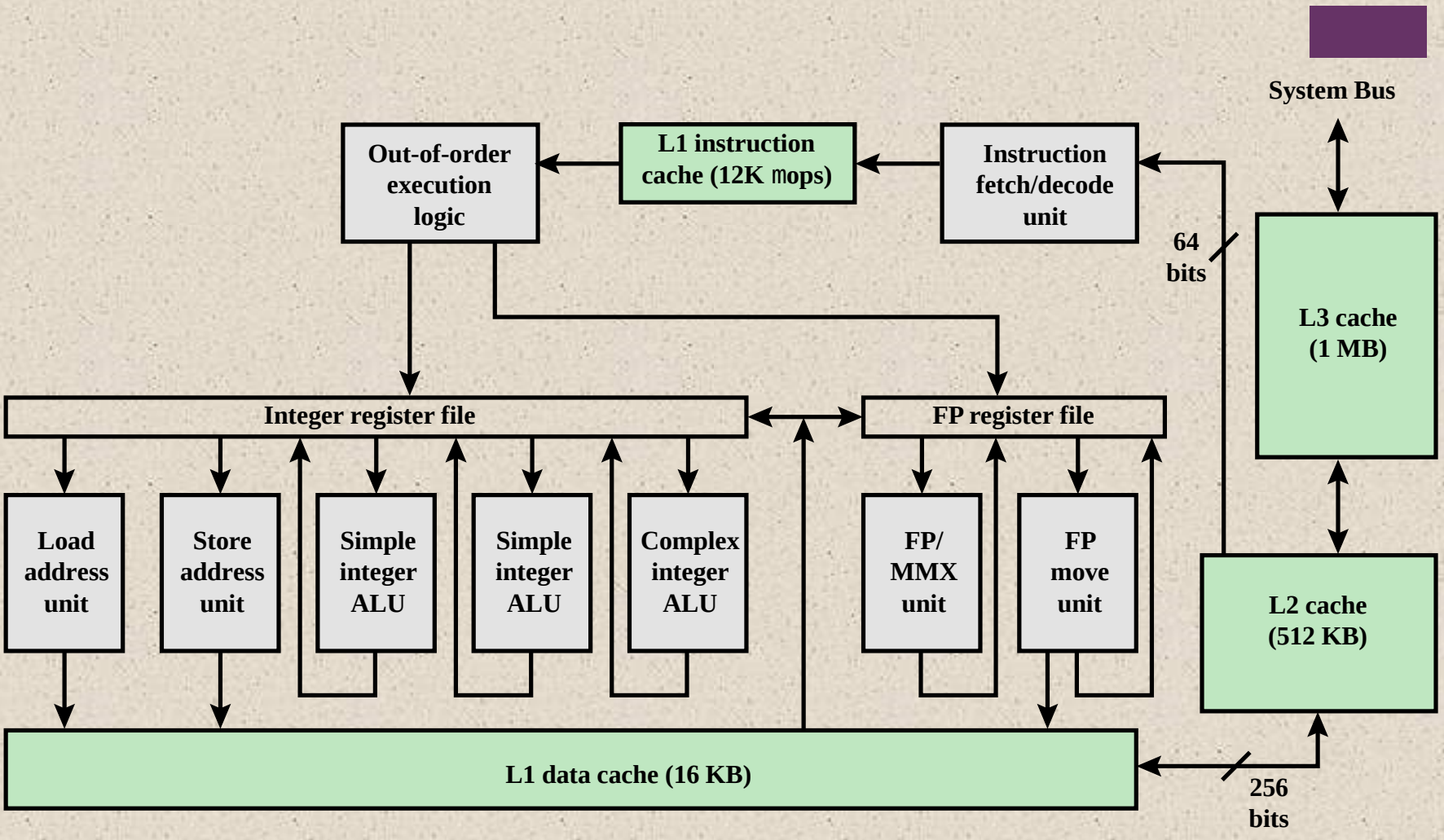


Figure 4.18 Pentium 4 Block Diagram

Şekil 4.18, üç önbelleğin yerleşimini vurgulayarak Pentium 4 organizasyonunun basitleştirilmiş bir görünümünü sağlar.

- İşlemci çekirdeği dört ana bileşenden oluşur:

- **Fetch/decode unit:**
  - Program komutlarını sırayla L2 önbelleğinden alır,
  - bunları bir dizi mikro işlem olarak çözer ve
  - sonuçları L1 komut önbelleğinde depolar.

- **Out-of-order execution logic:**
  - Veri bağımlılıklarına ve kaynak kullanılabilirliğine bağlı olarak mikro işlemlerin (*micro-operations*) yürütülmesini zamanlar; bu nedenle mikro işlemler, komut akışından getirilenden farklı bir sırada yürütülmek üzere programlanabilir.
  - Zaman izin verdiği ölçüde, bu birim gelecekte gerekli olabilecek mikro işlemlerin spekülatif olarak yürütülmesini planlar.

- **Execution units:**
  - Bu birimler mikro işlemleri yürütür, gerekli verileri L1 veri önbelleğinden alır ve sonuçları geçici olarak registerlarda depolar.

- **Memory subsystem:**
  - Bu birim, **L2** ve **L3 önbelleklerini** ve L1 ve L2 önbelleklerinde «*cache miss*» olduğunda ana belleğe erişmek ve sistem G / Ç kaynaklarına erişmek için kullanılan **sistem veri yolunu** içerir.

Table 4.5 Pentium 4 Cache Operating Modes

Control Bits		Operating Mode		
CD	NW	Cache Fills	Write Throughs	Invalidates
0	0	Enabled	Enabled	Enabled
1	0	Disabled	Enabled	Enabled
1	1	Disabled	Disabled	Disabled

Note: CD = 0; NW = 1 is an invalid combination.

L1 veri önbelleği, CD (cache disable) ve NW (not write-through) bitleri (Tablo 4.5) olarak etiketlenen kontrol registerlarının birindeki iki bit tarafından kontrol edilir.

Summary

Chapter 4

- Computer memory system overview
  - Characteristics of Memory Systems
  - Memory Hierarchy
- Cache memory principles
- Pentium 4 cache organization

Cache Memory

- Elements of cache design
  - Cache addresses
  - Cache size
  - Mapping function
  - Replacement algorithms
  - Write policy
  - Line size
  - Number of caches