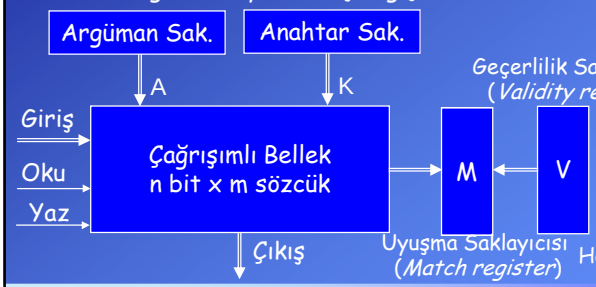
Çağrışımlı Bellek, İçerikle Adreslenen Bellek (*Associative Memory, Content Addressable Memory* – CAM)

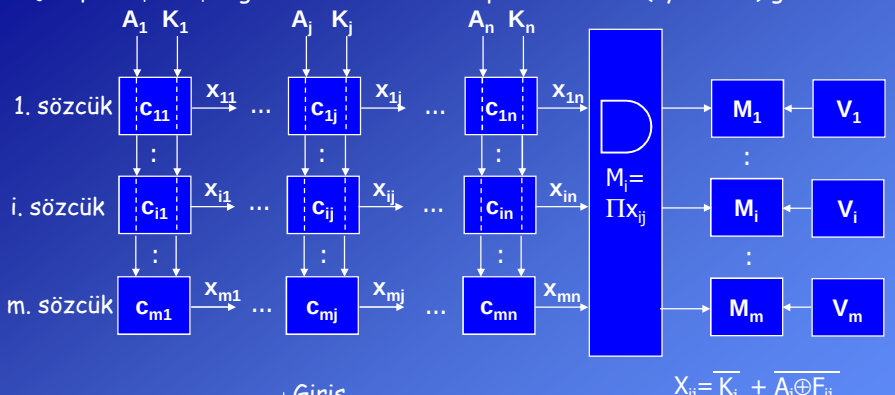
Cep bellek konusuna geçmeden önce cep bellek sistemlerinde kullanılan çağrışımlı bellek tanıtılacaktır.

- Hızlı arama yapabilmek için kullanılır.
- Bellek ve paralel arama için kullanılan sayısal devrelerden oluşur.
- Girişine verinin kendisi (Argüman – A) ve bu verinin ne kadarlık kısmının aramada kullanılacağı (Anahtar *Key*-K) verilir.
- Eğer aranan veri bellekte varsa uyuşma saklayıcısının ilgili biti "1" olur, bellekteki veri çıkıştan alınabilir.
- İlk elektrik verildiğinde bellekte rasgele değerler olacağından geçerlilik (*valid*) bitlerine de gerek duyulur. Başlangıçta V=0'dır. i. satıra veri yazılınca V_i=1 yapılır.

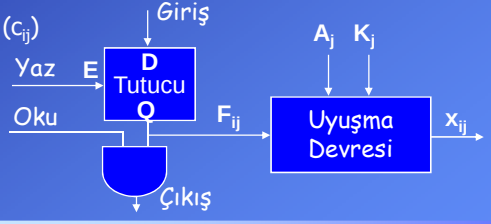


Örnek:
A: 1011 1111
K: 1111 0000
1100 1111 M=0
1011 1010 M=1
Çıkış: 1011 1010

İç Yapı: A_i ve K_i değerleri tüm hücrelere paralel olarak (aynı anda) gider.



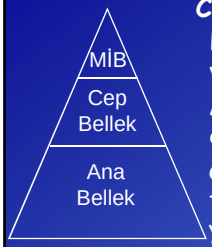
Bir hücrenin (c_{ij}) yapısı:



$$X_{ij} = \overline{K_j} + \overline{A_j \oplus F_{ij}}$$
$$X_{ij} = \overline{K_j} + A_j \odot F_{ij}$$
$$X_{ij} = \overline{K_j} + A_j \cdot F_{ij} + \overline{A_j} \cdot \overline{F_{ij}}$$

$$M_i = V_i \cdot \prod_{j=1}^{n} x_{ij}$$

Cep (Ön) Bellek Sistemi (*Cache Memory*)



Bellek hiyerarşisindeki cep bellek ve ana bellek MİB'in program ve veriler için doğrudan erişebildiği sistem içi belleklerdir.

Ana bellek: DRAM (*Dynamic Memory*) olarak üretilir. Kapasitesi daha büyük, birim maliyeti daha düşük, ancak daha yavaştır.

Cep bellek: SRAM (*Static Memory*) olarak üretilir. İşlemci ile aynı tümdevrede olabilir. Kapasitesi daha küçük, birim maliyeti daha yüksek, ancak daha hızlıdır.

Amaç: Farklı belleklerden uygun miktarlarda kullanarak ve sık erişilen verileri daha hızlı olan belleğe yerleştirerek maliyeti düşük ve ortalama erişim süresi kısa bir bellek sistemi oluşturmaktır.

Veriler belleklere yerleştirilirken programlardaki **başvuru yöreselliği** özelliğinden yararlanılır.

Başvuru yöreselliği (*Locality of Reference*)

- **Coğrafi (Uzayda) Yöresellik (*Spatial*):** Belleğe bir başvuru yapıldıktan sonra büyük olasılıkla bir sonraki başvuru yakın bir adrese olacaktır.
- **Zamanda Yöresellik (*Temporal*):** Bir adrese başvurulduktan sonra büyük olasılıkla bir süre sonra aynı adrese tekrar başvurulur.

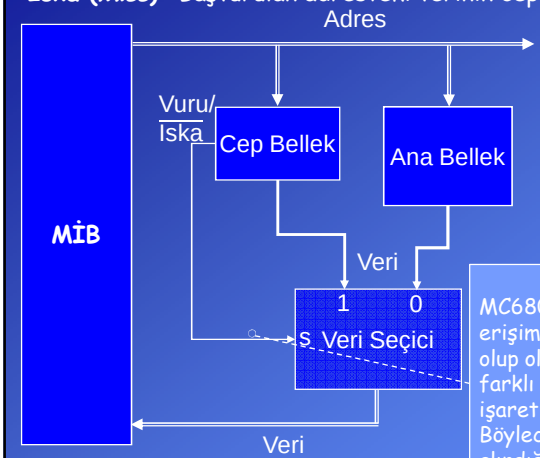
Yöreselliğin nedenleri: Programların sırasallığı, döngüler, diziler, tablolar.

Cep Bellek Sisteminin Çalışma Mantığı

Sık başvurulan adreslerdeki verilerin cep bellekte tutulması amaçlanır.

Vuru (*Hit*): Başvurulan adresteki verinin cep bellekte bulunması.

Iska (*Miss*): Başvurulan adresteki verinin cep bellekte bulunmaması.



Örnek:
Cep bellek erişim süresi: 20 ns.
Ana bellek erişim süresi: 100 ns.
Vuru oranı: H=0.9. %90 vuru var.
Ortalama bellek erişim süresi:
t_a= 0.9*20 + 0.1*100 = 28 ns.

MC68000 gibi asenkron bellek erişimi yapan sistemlerde, vuru olup olmamasına göre MİB'e farklı gecikmelerle DTACK işareti gönderilir.
Böylece veri cep bellekten alındığında yol çevriminin daha çabuk tamamlanması sağlanır.

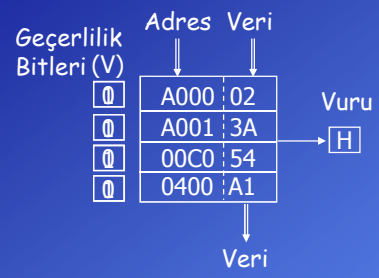
Cep Bellek Erişim Yöntemleri (*Mapping*)

1. Çağrışımlı (*Associative*) Erişim Yöntemi

a) Blok Yapısı Olmadan:

Gerçekte blok yapısı ile birlikte kullanılır.

En çok başvurulan adresler ve içerikleri bir çağrışımlı bellekte (*Associative memory*) tutulur.



Cep bellek dolduğunda bir yer değiştirme algoritması (*replacement*) kullanılarak cep bellekteki bir veri kaldırılır onun yerine yenisi getirilir.

FIFO (*First In First Out*): Cep belleğe önce gelmiş olan veri cep bellekten kaldırılır.

LRU (*Least Recently Used*): En az başvurulmuş olan veri cep bellekten kaldırılır.

Bu yöntemde yaşlanma sayaçları kullanılır.

LRU 11. bölümde açıklanmıştır.

Blok yapısı kullanılmadığında sadece zamanda yöresellikten yararlanılır, coğrafi yöresellikten yararlanılmamış olur.

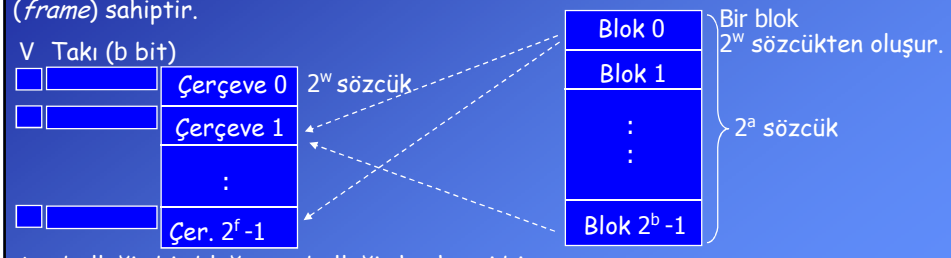
Bu nedenle bu yöntem gerçekte blok yapısı ile birlikte kullanılır.

b) Blok Yapılı Tam Çağrışımlı Erişim Yöntemi (*Full Associative*)

Coğrafi yöresellikten de yararlanabilmek için veriler cep belleğe bloklar halinde getirilir.

Cep bellek yönetim sistemi MİB'ten gelen adresi iki alt alana ayırarak değerlendirir:

Cep bellek, boyutuna bağlı olarak belli sayıda blok içerebilecek çerçeveye (*frame*) sahiptir.



Ana belleğin bir bloğu cep belleğin herhangi bir çerçevesinde yer alabilir.

Ana bellek 2^b adet bloktan oluşur.

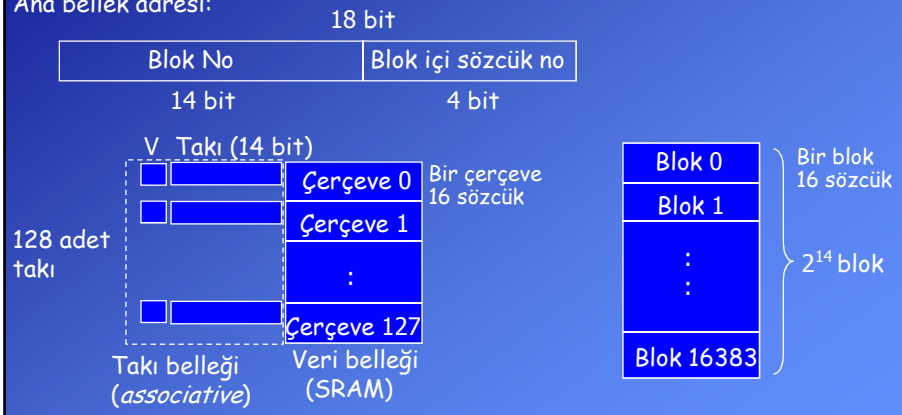
Bir çerçevede hangi bloğun olduğu takı (*tag*) bilgisinden anlaşılır. Bu yöntemde takı bilgisi olarak ana bellek adresinin blok numarası kullanılır.

Örnek (*Full Associative*):

Ana Bellek: 256K x sözcük
Blok boyu: 16 sözcük
Cep Bellek: 2K x sözcük
veri taşıyabiliyor.

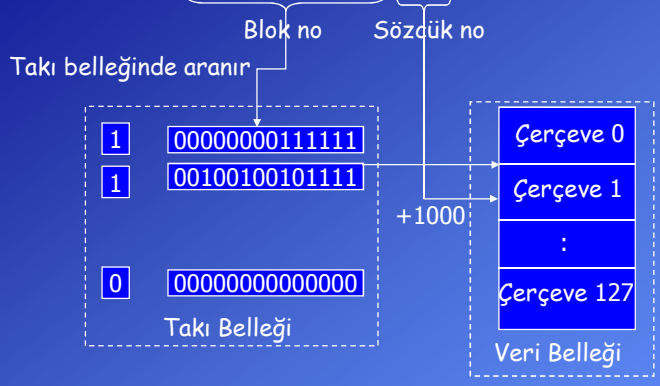
Adres: a = 18 bit
w = 4 bit Ana bellekte 2^{14} adet blok vardır. b = 14
Cep bellekte 2^{7} (128) adet çerçeve vardır. f = 7

Ana bellek adresi:



Örnek (devam):

MİB'ten gelen adres: 00 1001 0010 1111 1000



Eğer cep bellek dolduğunda blok değiştirme (*replacement*) yöntemi olarak **LRU** (*Least Recently Used*) kullanılıyorsa takı belleğinde her çerçeveye ait yaşlanma sayaçları da (*aging counter*) bulunur.

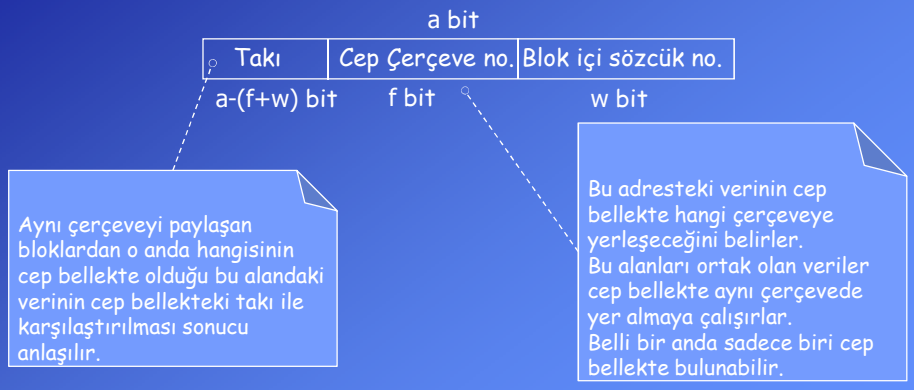
Bir çerçeveye başvurulmadığı zaman o çerçevenin sayacı arttırılır.

Blok değiştirme gerektiğinde "en yaşlı" çerçevedeki blok kaldırılır.

2. Doğrudan Dönüşüm (*Direct Mapping*)

Ana bellekteki bir bloğun cep bellekteki hangi çerçeveye yerleşebileceği belli ve sabittir.
Bloğun yeri cep içinde aranmadığından çağrışımlı (*associative*) bellek kullanılmaz.
Sadece o blok o anda cep bellekte var mı, yok mu araştırılır.

Cep bellek yönetim sistemi MİB'ten gelen adresi üç alt alana ayırarak değerlendirir:



Ana belleğin bir bloğu, sadece kendi adresindeki "cep çerçeve no" alanının belirlediği çerçevede yer alabilir.

Doğrudan dönüşüm yönteminde her bloğun cepteki yeri (çerçeve no) belli olduğundan,

- Cep bellekte boş çerçeveler olsa bile aynı çerçeveyi kullanan iki blok aynı anda cep bellekte yer alamaz.
- Blok değiştirme aşamasında karar verme yöntemlerine gerek yoktur.
- Çağrışımlı bellek kullanımına gerek yoktur.

Örnek (*Direct Mapping*):

Ana Bellek: 256K x sözcük
Blok boyu: 16 sözcük
Cep Bellek: 2K x sözcük
veri taşıyabiliyor.

Adres: a = 18 bit
w = 4 bit Ana bellekte 2^{14} adet blok vardır. b = 14
Cep bellekte 2^{7} (128) adet çerçeve vardır. f = 7

Ana bellek adresi:

18 bit		
Takı	Cep Çerçeve no	Blok içi sözcük no
7 bit	7 bit	4 bit

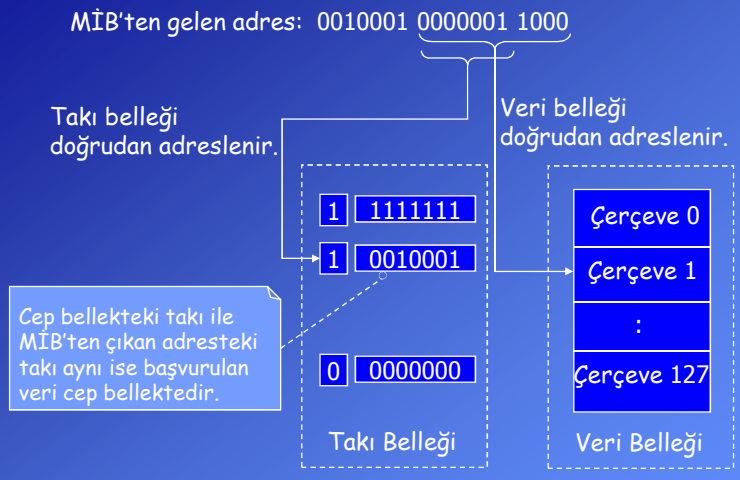
Örnek sistemde aşağıdaki iki adresteki veri cep bellekte aynı çerçeveye yerleşemeye çalışacaktır.

Takı	Çerçeve no	Sözcük no
0000000	0000000	XXXX
0000001	0000000	XXXX

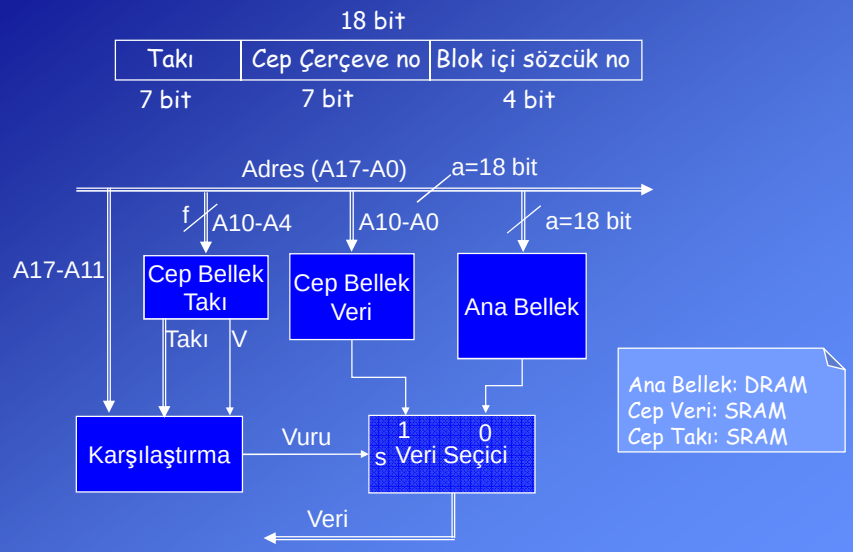
Her ikisinin de çerçeve numarası alanları 0000000 olduğundan Çerçeve 0'da aranacaklardır.

O anda hangisinin gerçekte cep bellekte olduğu adresteki takı alanı ile cep bellekteki takı alanının karşılaştırılmasıyla belirlenir.

Örnek (*Direct Mapping*):



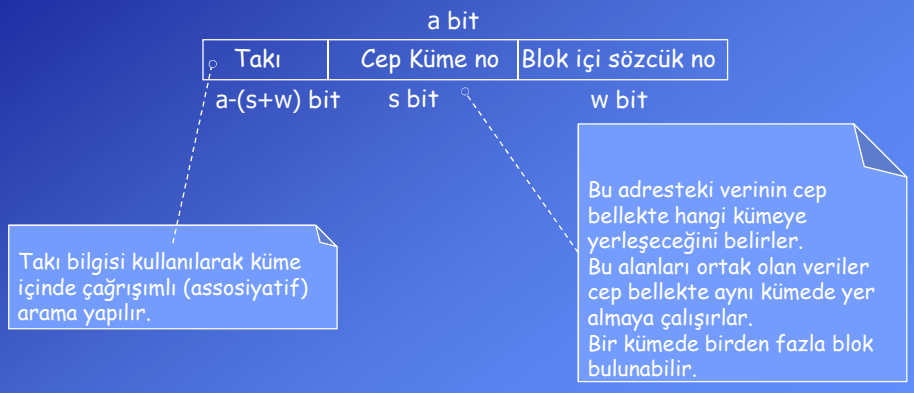
Örnek sistemin blok diyagramı:

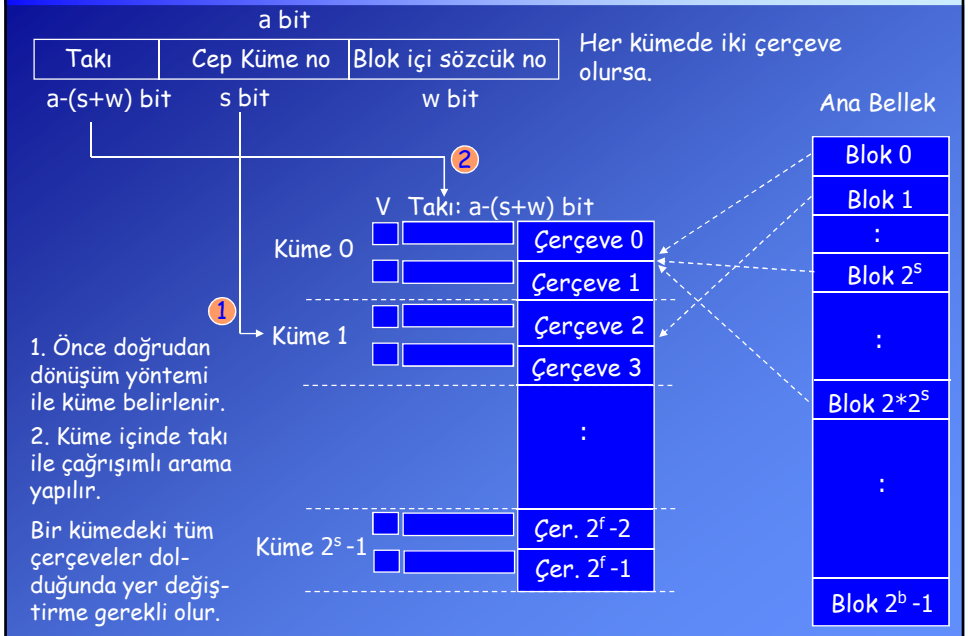


3. Kümeli ve Çağrışımlı (*Set Associative*) Dönüşüm

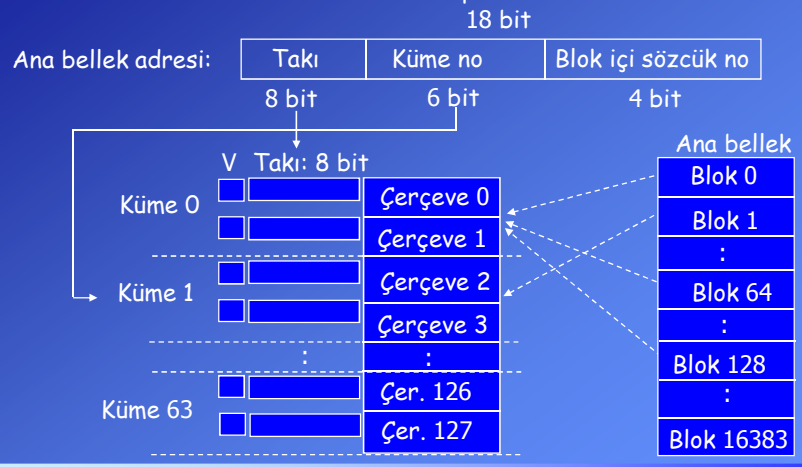
Doğrudan erişim ile tam çağrışımlı erişim arasında bir yöntemdir.
Cep belleğin kümlerden (*set*) oluştuğu ve her kümede belli sayıda çerçeve olduğu düşünülür.

Cep bellek yönetim sistemi MİB'ten gelen adresi üç alt alana ayırarak değerlendirir:





Örnek (*Set Assosciative*) :
Ana Bellek: 256K x sözcük
Blok boyu: 16 sözcük
Cep Bellek: 2K x sözcük
veri taşıyabiliyor.
Adres: a = 18 bit
w = 4 bit Ana bellekte 2^{14} adet blok vardır. b = 14
Cep bellekte 2^7 adet (128) çerçeve vardır. f = 7
Her kümede 2 blok olsun. Bu durumda cep bellekte 64 küme vardır.



Cep Bellek – Ana Bellek Etkileşimi

→ Okuma (Vuru): Veri cep bellekten okunur.

→ Okuma (Eğer ıska olursa):

• **Doğrudan Okuma (*Read Through – RT*):** Veri ana bellekten MİB'e okunurken aynı anda cebe de getirilir. Cep belleğe ve ana belleğe paralel erişilir.

• **Dolaylı Okuma (*No Read Through - NRT*):** Veri ana bellekten önce cep belleğe getirilir, sonra MİB cep belleği okur.

→ Yazma (Vuru Durumu):

• **Doğrudan Yazma (*Write Through – WT*):** Her yazma çevriminde veri hem cep belleğe hem de ana belleğe yazılır.

• **Sonradan Yazma (*Write Back –WB*):** Veriler sadece cep belleğe yazılır.

Değişikliğe uğrayan blok ancak cep bellekten kaldırılırken ana belleğe yazılır.

a. Basit sonradan yazma (*Simple Write Back – SWB*): Cep bellekten çıkartılan her blok ana belleğe yazılır. Zaman kaybettirir.

b. Bayraklı sonradan yazma (*Flagged Write Back – FWB*): Sadece değişikliğe uğramış olan bloklar cep bellekten çıkartılırken ana belleğe yazılır.

Değişen blokları belirleyebilmek için takı belleğinde "geçerlilik" bitleri ile birlikte bir de "**kirlenme**" (***dirty***) biti bulunur.

→ Yazma (Iska Durumu):

• **Cebe Yükleyerek Yazma (*Write Allocate - WA*):** Ana bellekte değiştirilen blok aynı zamanda cep belleğe de getirilir.

• **Cebe Yüklemeden Yazma (*No Write Allocate - NWA*):** Veri sadece ana belleğe yazılır.

Daha sonra eğer bu veri okunmak istenirse ıska olacağından ilgili blok cep belleğe getirilir.

Doğrudan yazma (WT) yöntemi cebe yükleyerek (WA) ya da yüklemeden yazma yöntemleri (NWA) ile birlikte kullanılabilir. WTWA, WTNWA

Sonradan yazma (WB) yönteminde cep belleği sürekli güncel tutmak için cebe yükleyerek (WA) yazma kullanılır. WBWA

Takı beleğinde bulunabilen bilgiler:

Geçerlilik (V) ve takı bilgisine ek olarak, kullanılan yöntemlere bağlı olarak takı belleğinde aşağıdaki bilgiler de bulunabilir:

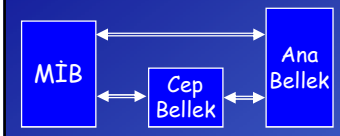
LRU kullanılıyorsa yaşlanma sayaçları, bayraklı sonradan yazma (*Flagged Write Back – FWB*) yöntemi kullanılıyorsa "**kirlenme**" (D) biti.

Takı beleğinin bir satırı:

MİB - Cep Bellek – Ana Bellek Bağlantısı

MİB - cep bellek – ana bellek bağlantıları iki farklı şekilde yapılabilir:

a) Paralel bağlantı



- ***Read Through* (RT):** Blok ana bellekten cep belleğe aktarılırken gerekli veri aynı zamanda MİB tarafından paralel olarak okunabilir.
- ***Load Through* (LT):** MİB cep belleğe veri yazarken veri aynı zamanda paralel olarak ana belleğe de yazılır.

Paralel yapı özellikle Doğrudan Yazma (*Write Through - WT*) yöntemi için uygundur. Sonradan Yazma (*Write Back -WB*) yöntemi ile de kullanılabilir.

b) Seri bağlantı

- *No* ***Read Through* (NRT):** Blok önce ana bellekten cep belleğe aktarılır ardından gerekli veri MİB tarafından cep bellekten okunur.
- ***No Load Through* (NLT):** MİB veriyi cep belleğe yazar, ardından veri cep bellekten ana belleğe aktarılır.

Seri yapı Sonradan Yazma (*Write Back -WB*) yöntemi için uygundur.

Erişim Süreleri:

t_a : Ortalama Bellek Erişim süresi (*Average memory access time*)
w : Yazma oranı (*Write ratio*) (yazma erişimleri sayısı / toplam erişim sayısı)
h : Vuru oranı (*Hit ratio*)
t_{cache} : Cep bellek erişim süresi (*Cache memory access time*)
t_{main} : Ana bellek erişim süresi (*Main memory access time*)
t_{trans}: Cep belleğe blok aktarım süresi (*Time to transfer block to cache*)
w_d: Blok değişme (kirlenme) olasılığı

	WT, RT/LT		WB,WA, NRT/NLT	
	NWA	WA	SWB	FWB
Okuma Vuru (1-w)h	t_{cache}	t_{cache}	t_{cache}	t_{cache}
Okuma Iska (1-w)(1-h)	t_{trans}	t_{trans}	$2t_{trans}+t_{cache}$	$w_d\,(2t_{trans}+t_{cache})$ + $(1-w_d)(t_{trans}+t_{cache})$
Yazma Vuru wh	t_{main}	t_{main}	t_{cache}	t_{cache}
Yazma Iska w(1-h)	t_{main}	$t_{main}+t_{trans}$	$2t_{trans}+t_{cache}$	$w_d\,(2t_{trans}+t_{cache})$ + $(1-w_d)(t_{trans}+t_{cache})$

Erişim Süreleri Hesabı:

- **Write Through with Write Allocate, Read/Load Through (WTWA, RT/LT):**

Okuma Vuru + Okuma Iska + Yazma Vuru + Yazma Iska

$t_a = (1-w)h\ t_{cache} + (1-w)(1-h)t_{trans} + w \cdot h \cdot t_{main} + w(1-h)(t_{main} + t_{trans})$

$t_a = (1-w)h\ t_{cache} + (1-h)t_{trans} + w \cdot t_{main}$

- **Write Through with No Write Allocate, Read/Load Through (WTNWA,RT/LT)**

$t_a = (1-w)h\ t_{cache} + (1-w)(1-h)t_{trans} + w \cdot h \cdot t_{main} + w(1-h)t_{main}$

$t_a = (1-w)\ t_{cache} + (1-w)(1-h)t_{trans} + w \cdot t_{main}$

- **Simple Write Back with Write Allocate,No Read Through (SWBWA,NRT/NLT)**

Okuma Vuru + Okuma Iska +Yazma Vuru + Yazma Iska

$t_a = (1-w)h\ t_{cache} + (1-w)(1-h)(2t_{trans} + t_{cache}) + w \cdot h \cdot t_{cache} + w(1-h)(2t_{trans} + t_{cache})$

$t_a = t_{cache} + (1-h) \cdot 2 \cdot t_{trans}$

t_{trans} terimlerinden biri cep bellekteki bloğu ana belleğe geri koymak için, diğeri de yeni bloğu ana bellekten cep belleğe getirmek içindir.

- **Flagged Write Back,Write Allocate, No Read Through (FWBWA,NRT/NLT):**

$t_a = t_{cache} + (1-h)t_{trans} + w_d \cdot (1-h)t_{trans}$

$t_a = t_{cache} + (1-h)(1+w_d)t_{trans}$

Cep bellek içeren örnek işlemciler:

- **Intel386™ ve öncesi:** Cep bellek işlemci tümdevresinin dışında SRAM bellek.
- **Intel486™ (1989)**
  8-KByte on-chip (L1)
- **Intel® Pentium® (1993)**
  L1 on-chip: 8 KB komut, 8 KB veri cebi (Harvard mimarisi)
- **Intel P6 Ailesi: (1995-1999)**
  - **Intel Pentium Pro:**
    L1 on-chip: 8 KB komut, 8 KB veri cebi (Harvard mimarisi)
    İlk defa bu yapıda L2 cep bellek işlemci ile aynı yapının içine tümleştirildi.
    L2 on-chip: 256 KB. L1 ve L2 cep belleklerin işlemci ile bağlantıları farklıdır.
  - **Intel Pentium II:**
    L1 on-chip: 16 KB komut, 16 KB veri cebi (Harvard mimarisi)
    L2 on-chip: 256 KB, 512 KB, 1 MB
- **Intel® Pentium® M (2003)**
  L1 on-chip: 32 KB komut, 32 KB veri cebi
  L2 on-chip: 2 MByte'a kadar

Intel® Core™ i7-980X Processor Extreme Edition (2010)
Tek tümdevrede çok işlemci var: 6 Çekirdek (core)
12 MB smartcache: Tüm çekirdekler kullanır.