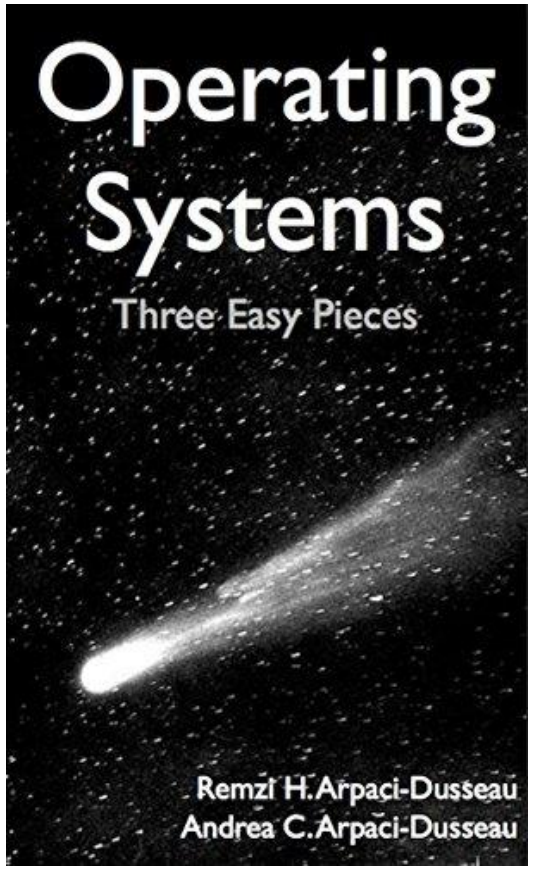
İşletim Sistemleri



6. Ders

Prof. Dr. Kemal Bıçakcı

Alıştırma Sorusu

Bir sayfalama sisteminde, sanal adresler 48-bit, fiziksel adresler ise 32-bit uzunluğundadır. Bu sistemde toplam kaç sayfa vardır?

a) 2^{16}

b) 2^{32}

c) 2^{48}

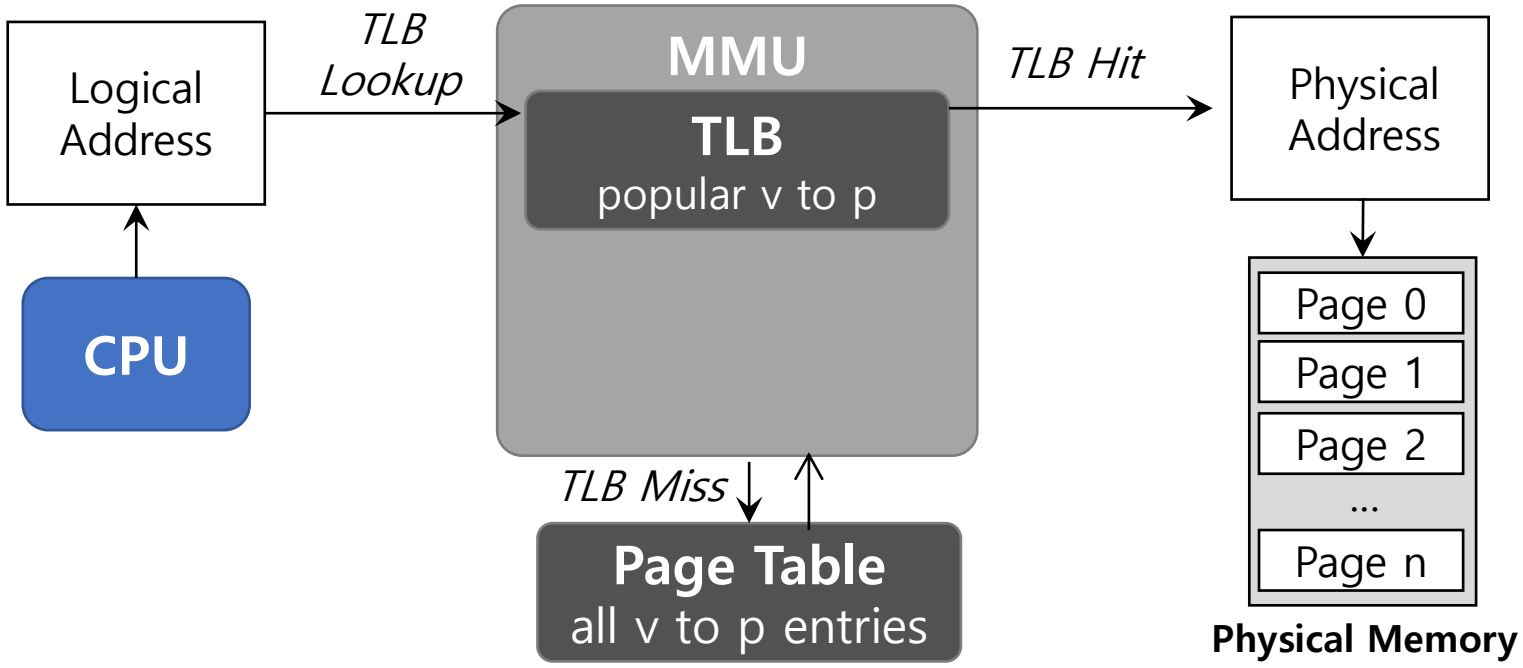
d) Verilen bilgiler hesaplama yapmak için yeterli değildir.

19. Adres Çevrim Önbellekleri (Translation Lookaside Buffers)

Operating System: Three Easy Pieces

TLB (Translation Lookaside Buffer)

- Bellek yönetim birimi (MMU) donanımının bir parçasıdır.
  - Sanal adres - fiziksel adres çevriminde popüler sayfalara ait donanım önbelleği.



Address Translation with MMU

TLB Çalışması - Temel Algoritma

```
1: VPN = (VirtualAddress & VPN_MASK ) >> SHIFT
2: (Success , TlbEntry) = TLB_Lookup(VPN)
3:    if(Success == True){ // TLB Hit
4:       if(CanAccess(TlbEntry.ProtectBit) == True ){
5:         offset = VirtualAddress & OFFSET_MASK
6:         PhysAddr = (TlbEntry.PFN << SHIFT) | Offset
7:         AccessMemory(PhysAddr)
8:       }else RaiseException(PROTECTION_ERROR)
```

- (1 lines) extract the virtual page number (VPN).
- (2 lines) check if the TLB holds the translation for this VPN.
- (5-8 lines) extract the page frame number from the relevant TLB entry, and form the desired physical address and access memory.

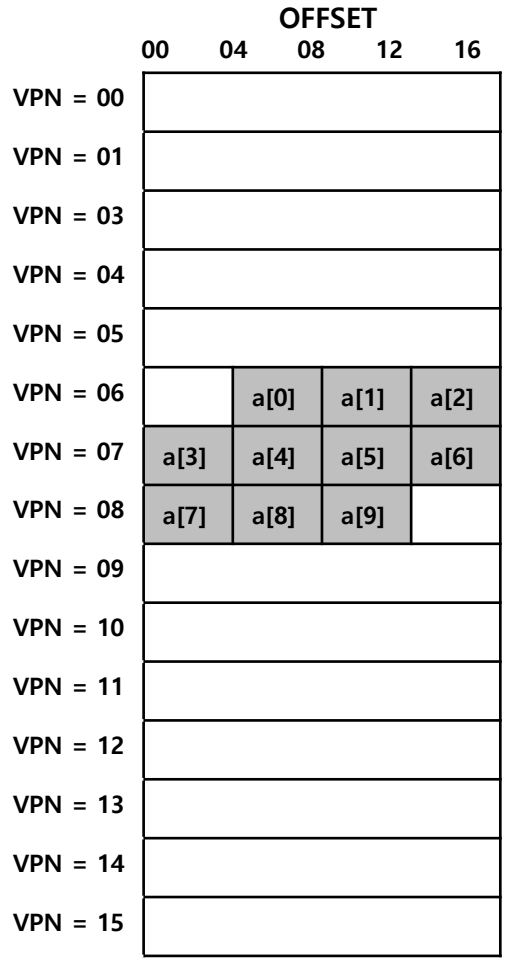
TLB Çalışması - Temel Algoritma (Devam)

```
11:     }else{ //TLB Miss
12:         PTEAddr = PTBR + (VPN * sizeof(PTE))
13:         PTE = AccessMemory(PTEAddr)
14:         (…)
15:     }else{
16:         TLB_Insert(VPN, PTE.PFN, PTE.ProtectBits)
17:         RetryInstruction()
18:     }
19:}
```

- (11-12 lines) The hardware accesses the page table to find the translation.
- (16 lines) updates the TLB with the translation.

Örnek: Bir Diziye Erişim

TLB, performansı nasıl artırır?



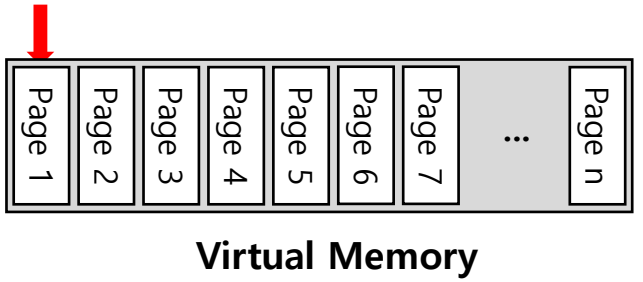

```
0:      int sum = 0 ;
1:      for( i=0; i<10; i++){
2:              sum+=a[i];
3:      }
```

TLB, mekansal yerellik (spatial locality) nedeniyle performansı artırır

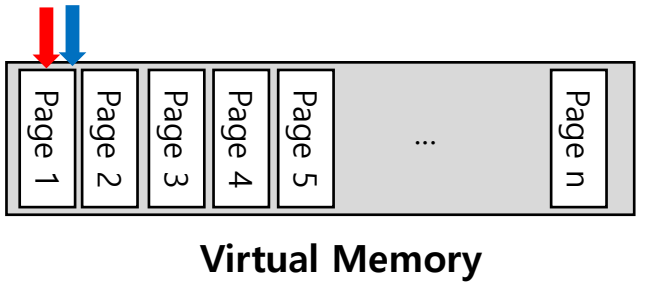
3 ıska ve 7 isabet.
Böylece TLB isabet oranı (hit rate) %70'tir.

Yerellik (Locality)

- Zamansal Yerellik (Temporal Locality)
  - Yakın zamanda erişilen bir buyruk veya veri öğesine, yakın gelecekte yeniden erişilmesi muhtemeldir.



- Mekansal Yerellik (Spatial Locality)
  - Bir program x adresindeki belleğe erişirse, büyük olasılıkla x adresi yakınındaki belleğe de erişecektir.



TLB Iskalarını Kim Ele Alır?

- CISC mimarisinde, TLB ıskalarını tamamen donanım ele alır.
- Donanım tarafından yönetilen TLB:
  - Donanım, sayfa tablolarının bellekte tam olarak nerede bulunduğunu bilmelidir.
  - Donanım, sayfa tablosunda doğru sayfa tablo girdisini bulur ve istenen çeviriyi yapar, güncelleme ve yeniden deneme buyruklarını çalıştırır.

TLB Iskalarını Kim Ele Alır? (Devam)

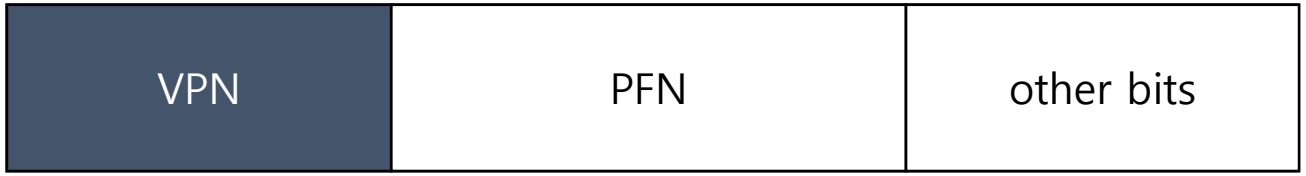
- RISC mimarisi yazılım tarafından yönetilen bir TLB'ye sahiptir.
- Bir TLB hatasında, donanım «exception» oluşturur (trap handler).
- Trap handler: İşletim sistemi içinde TLB ıskalarını işlemek amacıyla yazılmış kod.

TLB Kontrol Akış Algoritması (TLB Iskalarını İS ele aldığında)

```
1:        VPN = (VirtualAddress & VPN_MASK) >> SHIFT
2:        (Success, TlbEntry) = TLB_Lookup(VPN)
3:        if (Success == True) // TLB Hit
4:                if (CanAccess(TlbEntry.ProtectBits) == True)
5:                        Offset = VirtualAddress & OFFSET_MASK
6:                        PhysAddr = (TlbEntry.PFN << SHIFT) | Offset
7:                        Register = AccessMemory(PhysAddr)
8:                else
9:                        RaiseException(PROTECTION_FAULT)
10:       else // TLB Miss
11:                RaiseException(TLB_MISS)
```

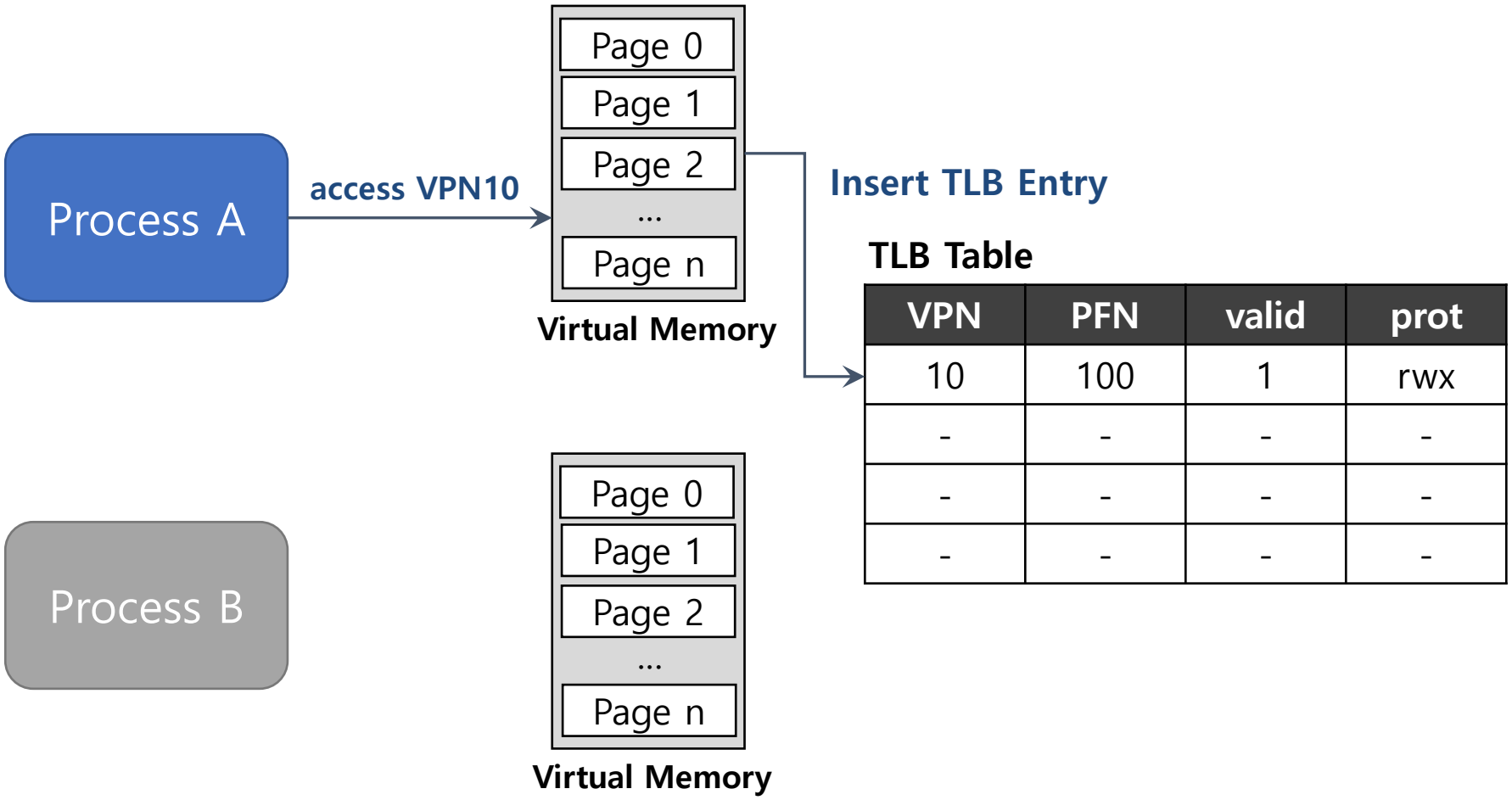
TLB satırı

- TLB, Tam İlişkilendirme (**Full Associative**) yöntemiyle yönetilir.
- Tipik bir TLB’nin 32,64 veya 128 girdisi olabilir.
- Donanım, istenen çevrimi bulmak için tüm TLB'yi paralel olarak arar.
- Diğer bitler: geçerli bitler, koruma bitleri, adres alanı tanımlayıcısı, kirli bit



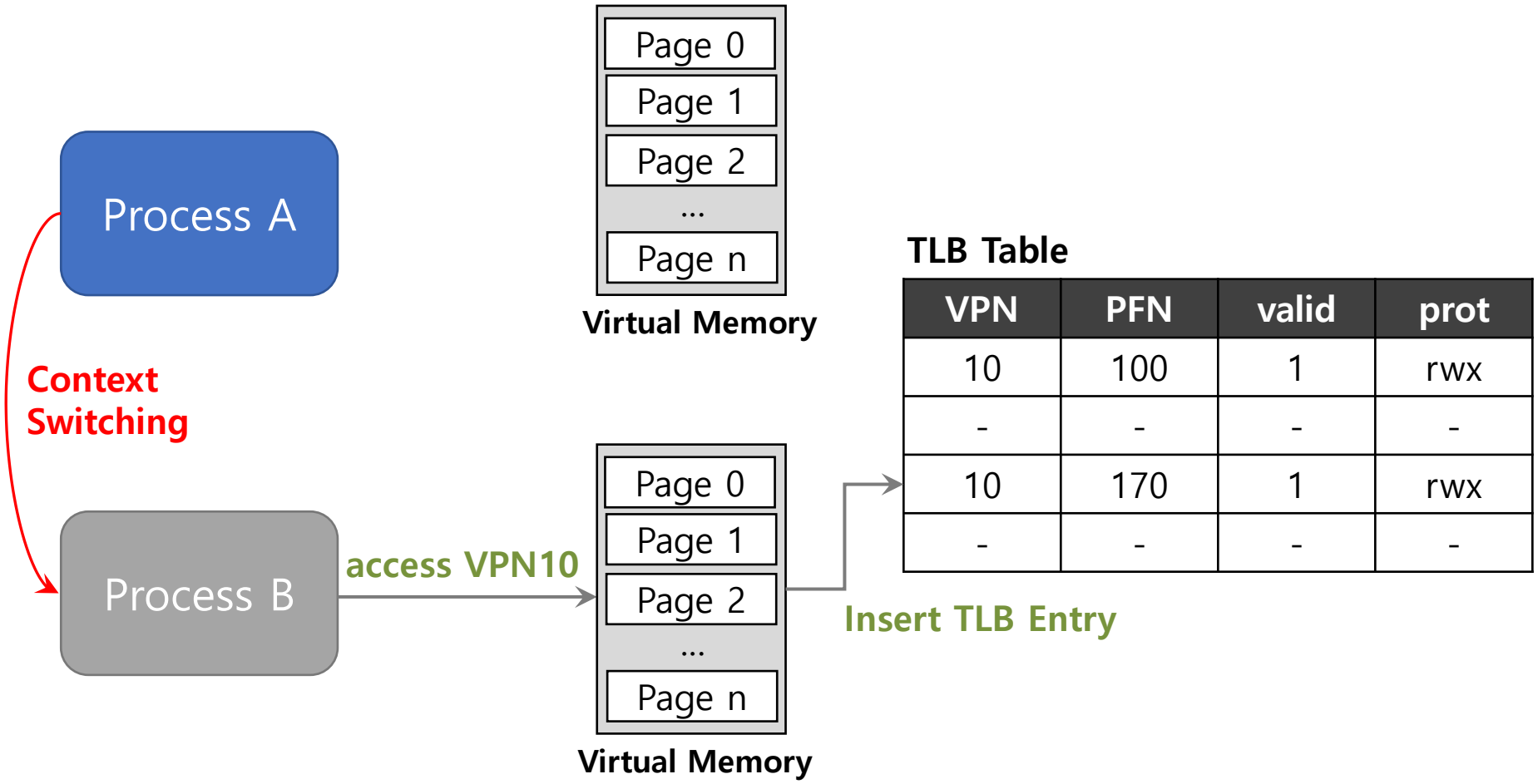
Typical TLB entry

Context Switching olduğunda TLB



VPN	PFN	valid	prot
10	100	1	rwx
-	-	-	-
-	-	-	-
-	-	-	-

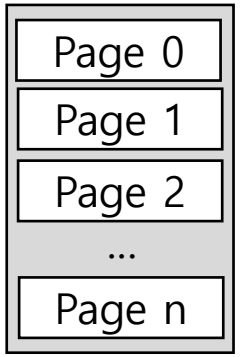
Context Switching olduğunda TLB (Devam)



VPN	PFN	valid	prot
10	100	1	rwx
-	-	-	-
10	170	1	rwx
-	-	-	-

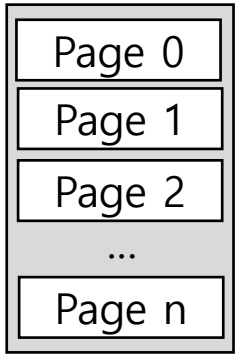
Context Switching olduğunda TLB (Devam)

Process A



Virtual Memory

Process B



Virtual Memory

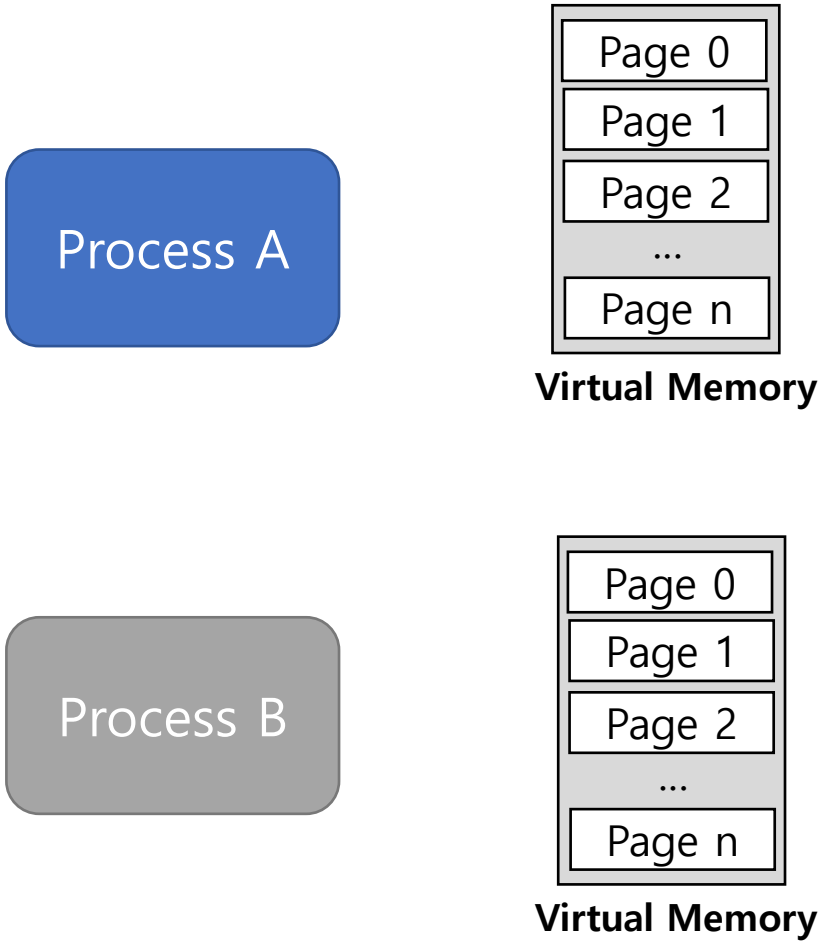
TLB Table

VPN	PFN	valid	prot
10	100	1	rwx
-	-	-	-
10	170	1	rwx
-	-	-	-

Hangi girdinin hangi işlem için olduğu ayırt edilemez

Problemin Çözümü

- TLB'de bir adres alanı tanımlayıcısı (ASID) alanı bulunur.



TLB Table

VPN	PFN	valid	prot	ASID
10	100	1	rwx	1
-	-	-	-	-
10	170	1	rwx	2
-	-	-	-	-

Paylaşım Durumu

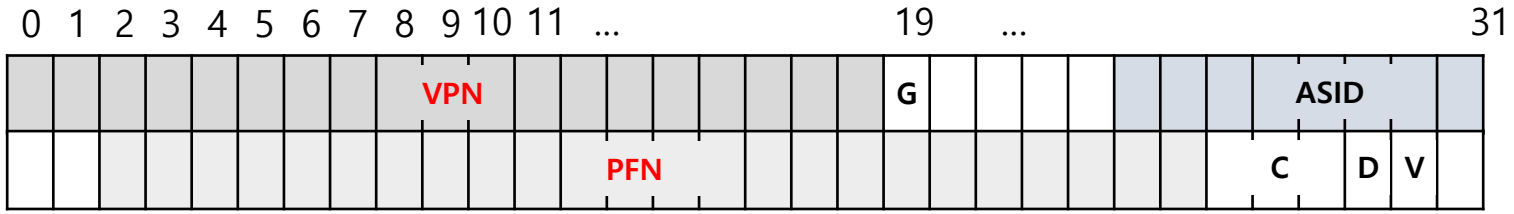
- İki işlem bir sayfayı paylaşır.
- İşlem 1, fiziksel sayfa 101'i İşlem 2 ile paylaşıyor.
- P1, bu sayfayı adres alanının 10. sayfasına eşler.
- P2, bu sayfayı adres alanının 50. sayfasına eşler.

VPN	PFN	valid	prot	ASID
10	101	1	rwx	1
-	-	-	-	-
50	101	1	rwx	2
-	-	-	-	-

Sayfaların paylaşılması, kullanımdaki fiziksel sayfa sayısını azalttığı için yararlıdır.

Gerçek bir TLB Girdisi

Bir TLB girdisine ait 64 bitin tümü (MIPS R4000 örneği)



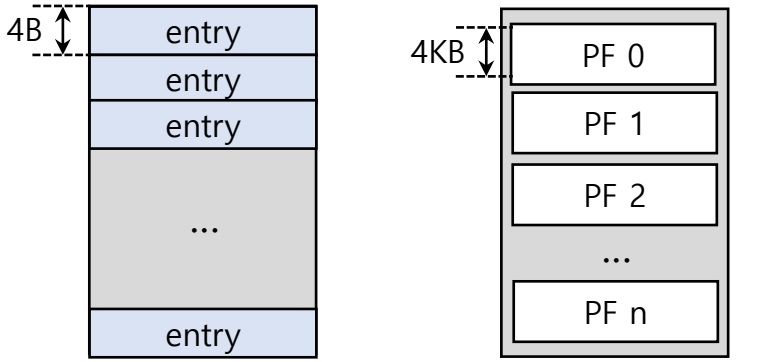
Flag	Content
19-bit VPN	The rest reserved for the kernel.
24-bit PFN	Systems can support with up to 64GB of main memory($2^{24} * 4KB$ pages).
Global bit(G)	Used for pages that are globally-shared among processes.
ASID	OS can use to distinguish between address spaces.
Coherence bit(C)	determine how a page is cached by the hardware.
Dirty bit(D)	marking when the page has been written.
Valid bit(V)	tells the hardware if there is a valid translation present in the entry.

20. Sayfalama: Daha Küçük Tablolar

Operating System: Three Easy Pieces

Sayfalama: Lineer Tablolar

- Sistemdeki her işlem için genellikle ayrı bir sayfa tablomuz olur.
- 4 KB sayfaları ve 4 baytlık sayfa tablosu girdileri olan 32 bitlik bir adres uzayı olduğunu varsayalım.

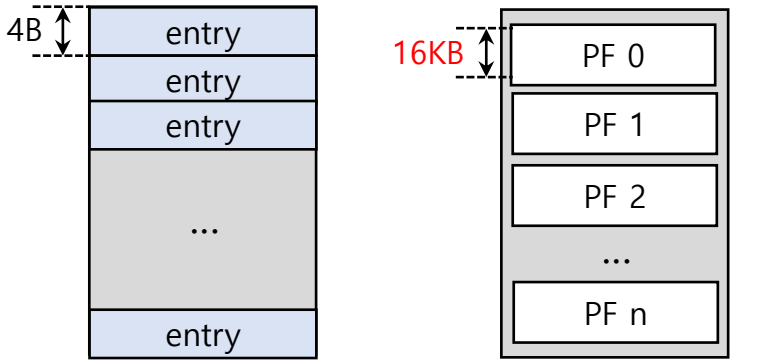


Sayfa Tablo Büyüklüğü = $\frac{2^{32}}{2^{12}} * 4B = 4MB$

Sayfa tabloları çok büyük ve bu nedenle çok fazla bellek tüketiyor.

Sayfalama: Daha Küçük Tablolar

- Sayfalar 4KB yerine 64KB olsun.



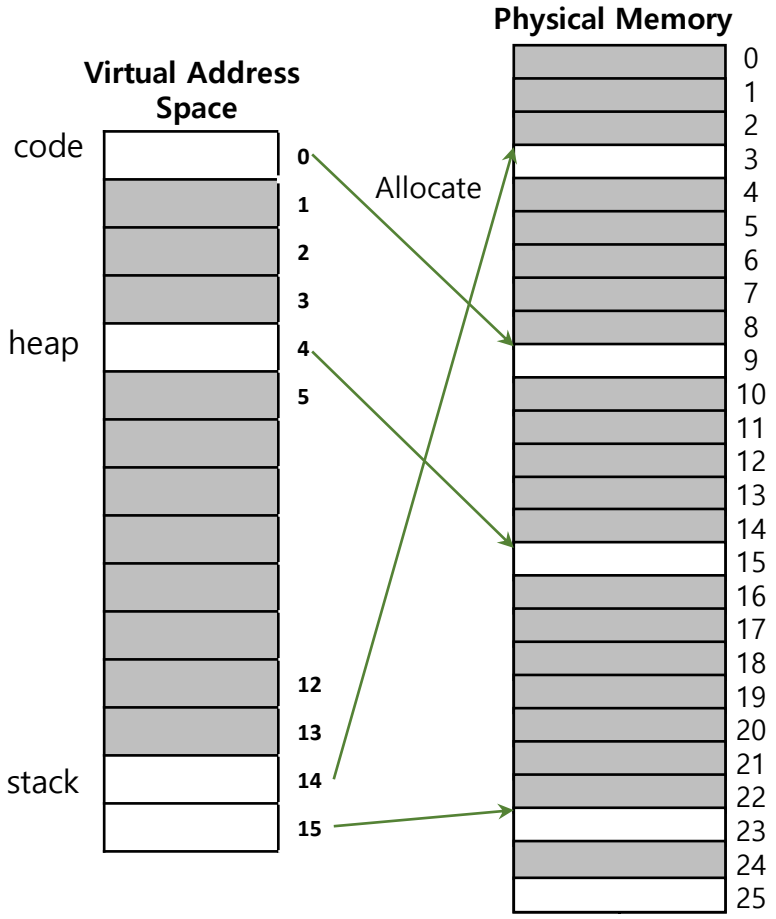
İşlem A'nın Sayfa Tablosu **Fiziksel Bellek**

$\frac{2^{32}}{2^{16}} * 4 = 0,25MB$ **(her sayfa tablosu)**

Büyük sayfalar dahili parçalanmaya yol açar.

Problem

Sayfa tablosu, işlemin adres uzayındaki tüm sayfalar için fiziksel belleği kullanıyor.



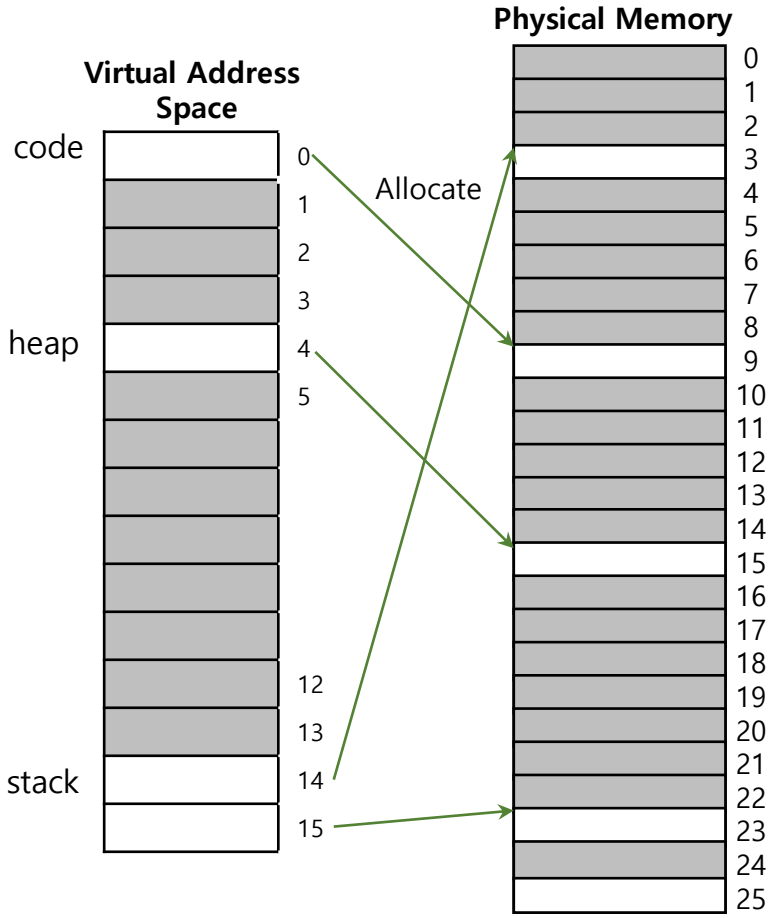
A 16KB Address Space with 1KB Pages

PFN	valid	prot	present	dirty
10	1	r-x	1	0
-	0	-	-	-
-	0	-	-	-
-	0	-	-	-
15	1	rw-	1	1
...	...	...	...	...
-	0	-	-	-
3	1	rw-	1	1
23	1	rw-	1	1

A Page Table For 16KB Address Space

Problem (Devam)

Sayfa tablosunun çoğu kullanılmamış, geçersiz girdilerle dolu.



A 16KB Address Space with 1KB Pages

PFN	valid	prot	present	dirty
10	1	r-x	1	0
-	0	-	-	-
-	0	-	-	-
-	0	-	-	-
15	1	rw-	1	1
...	...	...	...	...
-	0	-	-	-
3	1	rw-	1	1
23	1	rw-	1	1

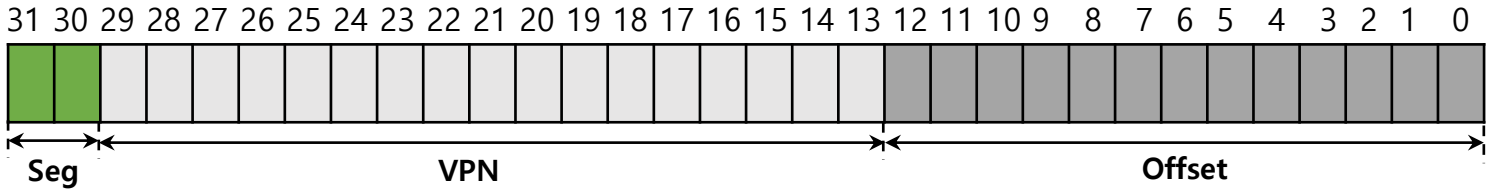
A Page Table For 16KB Address Space

Hibrit Yaklaşım: Sayfalama ve Bölütler

- Sayfa tablolarının bellek yükünü azaltmak için:
  - Taban yazmacını (Base register) bölütün kendisine işaret etmek için değil, o bölütün sayfa tablosunun fiziksel adresini tutmak için kullanmak.
  - Sınır yazmacı (Bound register) da sayfa tablosunun sonunu belirtmek için kullanılır.

Hibrit Yaklaşıma Basit Bir Örnek

- Her işlemin kendisiyle ilişkilendirilmiş üç sayfa tablosu vardır.
- İşlem çalışırken, bu bölütlerin her biri için taban yazmacı, o bölüt için bir doğrusal sayfa tablosunun fiziksel adresini içerir.



32-bit Virtual address space with 4KB pages

Seg value	Content
00	unused segment
01	code
10	heap
11	stack

Hibrit Yaklaşımda TLB Iskası

- Donanım, sayfa tablosunun fiziksel adresini öğrenir.
- Donanım, hangi taban ve sınır çiftinin kullanılacağını belirlemek için segment bitlerini (SN) kullanır.
- Donanım daha sonra buradaki fiziksel adresi alır ve sayfa tablosu girdisinin (PTE) adresini oluşturmak için VPN ile birleştirir.

```
01:     SN = (VirtualAddress & SEG_MASK) >> SN_SHIFT
02:     VPN = (VirtualAddress & VPN_MASK) >> VPN_SHIFT
03:     AddressOfPTE = Base[SN] + (VPN * sizeof(PTE))
```

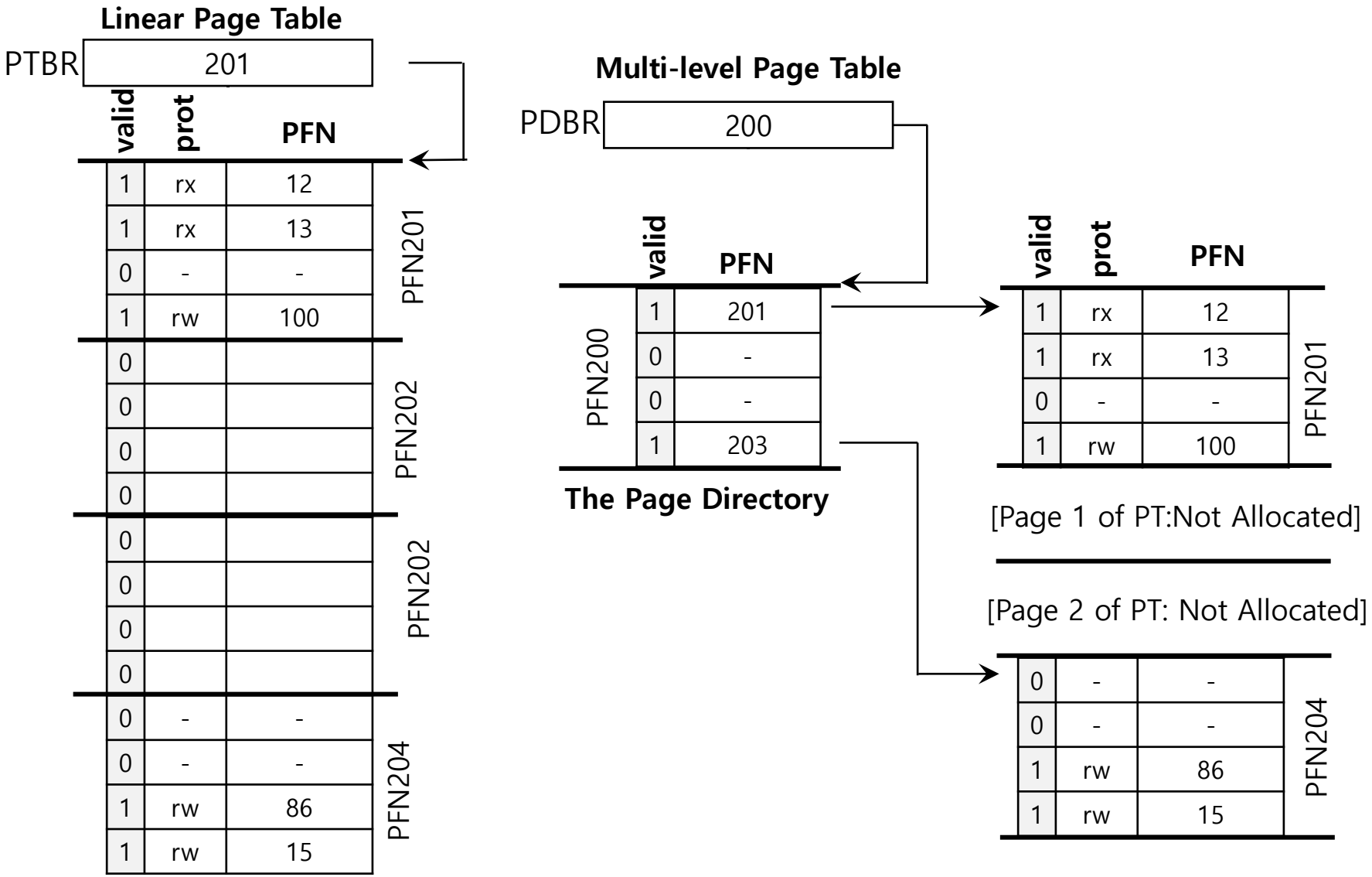
Hibrit Yaklaşımın Problemi

- Hibrit Yaklaşım sorunsuz değildir.
  - Büyük ama seyrek kullanılan bir yığınımız varsa, yine de çok fazla sayfa tablosu israfıyla karşılaşabiliriz.
  - Dış parçalanmanın yeniden ortaya çıkmasına neden olabilir.

Çok Seviyeli Sayfa Tabloları

- Doğrusal sayfa tablosunu ağaç yapısına dönüştürür.
- Sayfa tablosunu sayfa boyutunda birimlere ayırır.
- Sayfa tablosu girdilerinden oluşan bir sayfanın tamamı geçersizse, sayfa tablosunun o sayfasını hiç ayırmaz.
- Sayfa tablosunun bir sayfasının geçerli olup olmadığını izlemek için sayfa dizini (page directory) adı verilen yeni bir yapı kullanılır.

Çok Seviyeli Sayfa Tabloları: Sayfa Dizini



Lineer (Solda) ve Çok Seviyeli (Sağda) Sayfa Tabloları

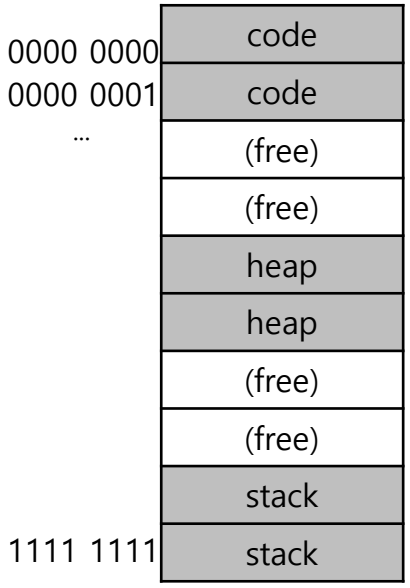
Çok Seviyeli Sayfa Tabloları: Sayfa dizini girdileri

- Sayfa dizini, sayfa tablosunun her sayfası için bir girdi içerir.
- Bir dizi sayfa dizini girdisinden (PDE) oluşur.
- PDE'nin bir geçerli biti ve sayfa çerçeve numarası (PFN) vardır.

Avantaj ve Dezavantajlar

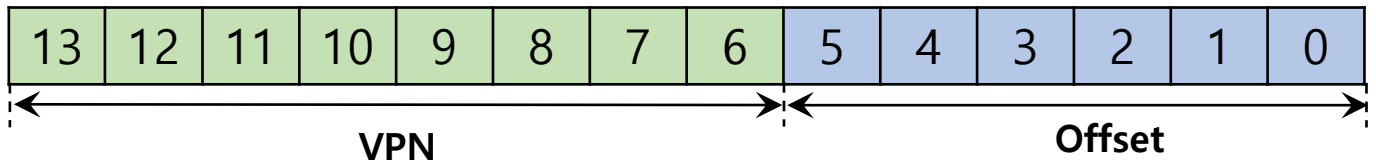
- Avantaj:
  - Yalnızca kullandığınız adres uzayı ile orantılı olarak sayfa tablosu alanı ayrılır.
  - İşletim sistemi, bir sayfa tablosu ayırması veya büyütmesi gerektiğinde herhangi bir sonraki boş sayfayı kullanabilir.
- Dezavantaj:
  - Verimsizlik (Çok seviyeli tablo, zaman-yer ödünleşmesine (time-space trade-off) küçük bir örnektir.
  - Karmaşıklık

Örnek



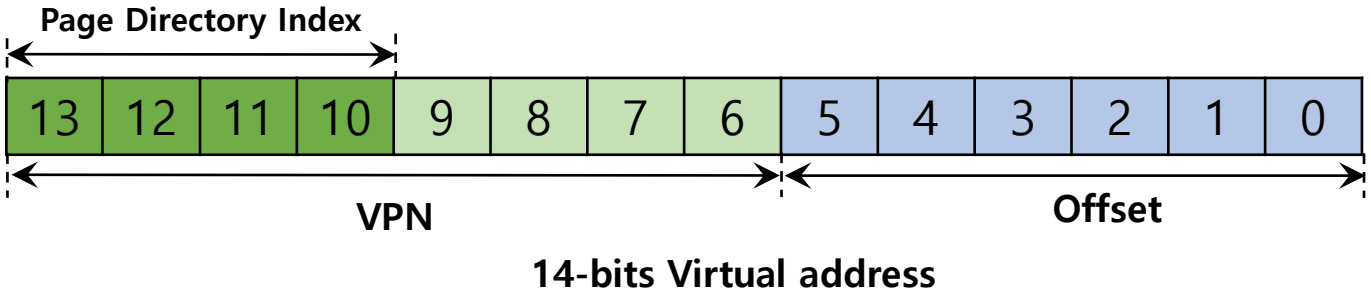
Flag	Detail
Address space	16 KB
Page size	64 byte
Virtual address	14 bit
VPN	8 bit
Offset	6 bit
Page table entry	$2^8(256)$

A 16-KB Address Space With 64-byte Pages



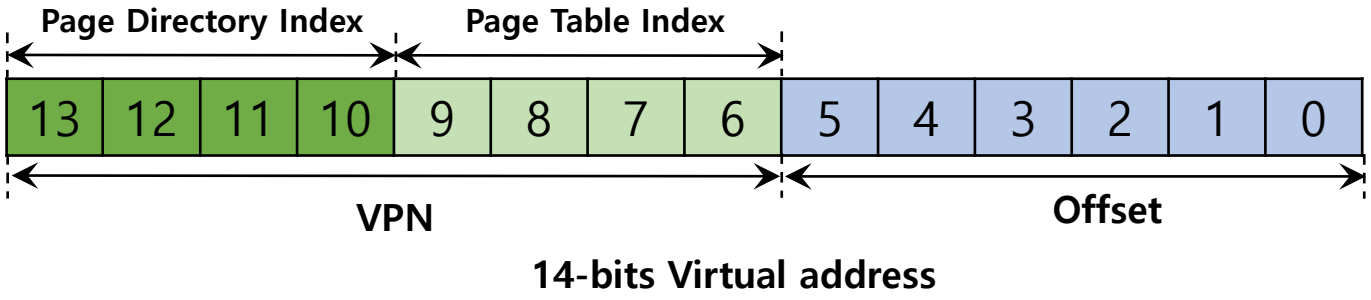
Örneğe Devam:
Sayfa Dizin İndeksi (Page Directory Index)

- Sayfa dizini, sayfa tablosunun her sayfası için bir girdiye ihtiyaç duyar.
- Bu örnekte 16 girdisi vardır.
- Sayfa dizini girdisi geçersiz ise «exception» oluşur (Erişim geçersiz).



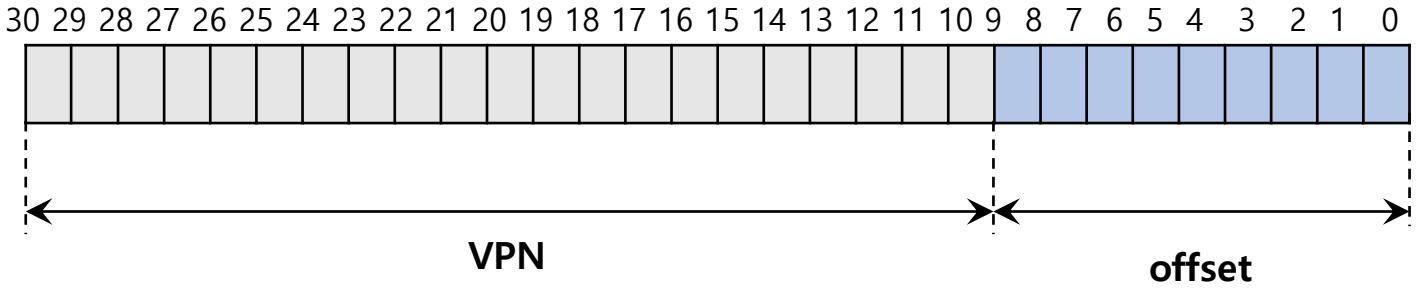
Örneğe Devam:
Sayfa Dizin İndeksi (Page Directory Index)

- PDE geçerli ise yapacak daha işimiz var demektir.
- Bu sayfa dizini girdisi tarafından işaret edilen sayfa tablosunun sayfasından Sayfa Tablosu İndeksi yardımı ile ilgili sayfa tablosu girdisi (PTE) bulunur.



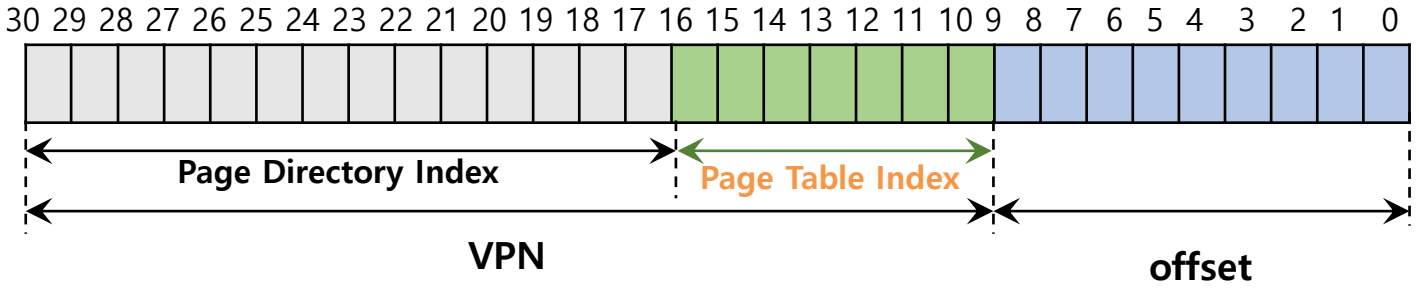
İki Seviyeden Fazlası Mümkün Mü?

- Bazı durumlarda, daha derin bir ağaç gerekli olur.



Flag	Detail
Virtual address	30 bit
Page size	512 byte
VPN	21 bit
Offset	9 bit

İki Seviyeden Fazlası Mümkün Mü?

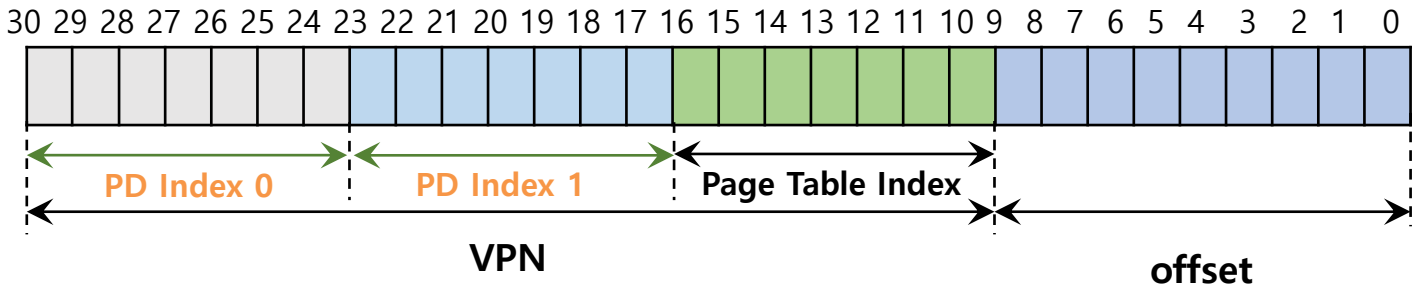


Flag	Detail
Virtual address	30 bit
Page size	512 byte
VPN	21 bit
Offset	9 bit
Page entry per page	128 PTEs

→ $\log_2 128 = 7$

İki Seviyeden Fazlası Mümkün Mü?

- Sayfa dizinimizde 2^14 girdi varsa, bunlar bir değil 128 sayfaya yayılır.
- Bu sorunu çözmek için, sayfa dizininin kendisini sayfa dizininin birden çok sayfasına bölerek ağacın daha ileri bir seviyesini oluşturabiliriz.



Çok Seviyeli Sayfa Tablosu Kontrol Akışı

```
01:     VPN = (VirtualAddress & VPN_MASK) >> SHIFT
02:     (Success,TlbEntry) = TLB_Lookup(VPN)
03:     if(Success == True)        //TLB Hit
04:       if(CanAccess(TlbEntry.ProtectBits) == True)
05:             Offset = VirtualAddress & OFFSET_MASK
06:             PhysAddr = (TlbEntry.PFN << SHIFT) | Offset
07:             Register = AccessMemory(PhysAddr)
08:       else RaiseException(PROTECTION_FAULT);
09:     else // perform the full multi-level lookup
```

- (1 lines) extract the virtual page number(VPN)
- (2 lines) check if the TLB holds the transalation for this VPN
- (5-8 lines) extract the page frame number from the relevant TLB entry, and form the desired physical address and access memory

Çok Seviyeli Sayfa Tablosu Kontrol Akışı

```
11:       else
12:                 PDIndex = (VPN & PD_MASK) >> PD_SHIFT
13:                 PDEAddr = PDBR + (PDIndex * sizeof(PDE))
14:                 PDE = AccessMemory(PDEAddr)
15:                 if(PDE.Valid == False)
16:                           RaiseException(SEGMENTATION_FAULT)
17:                 else // PDE is Valid: now fetch PTE from PT
```

- (11 lines) extract the Page Directory Index(PDIndex)
- (13 lines) get Page Directory Entry(PDE)
- (15-17 lines) Check PDE valid flag. If valid flag is true, fetch Page Table entry from Page Table

Çevrim Sürecini Hatırlayalım: TLB'yi Unutmayalım

```
18:     PTIndex = (VPN & PT_MASK) >> PT_SHIFT
19:     PTEAddr = (PDE.PFN << SHIFT) + (PTIndex * sizeof(PTE))
20:     PTE = AccessMemory(PTEAddr)
21:     if(PTE.Valid == False)
22:             RaiseException(SEGMENTATION_FAULT)
23:     else if(CanAccess(PTE.ProtectBits) == False)
24:             RaiseException(PROTECTION_FAULT);
25:     else
26:             TLB_Insert(VPN, PTE.PFN , PTE.ProtectBits)
27:             RetryInstruction()
```

Ters Sayfa Tabloları

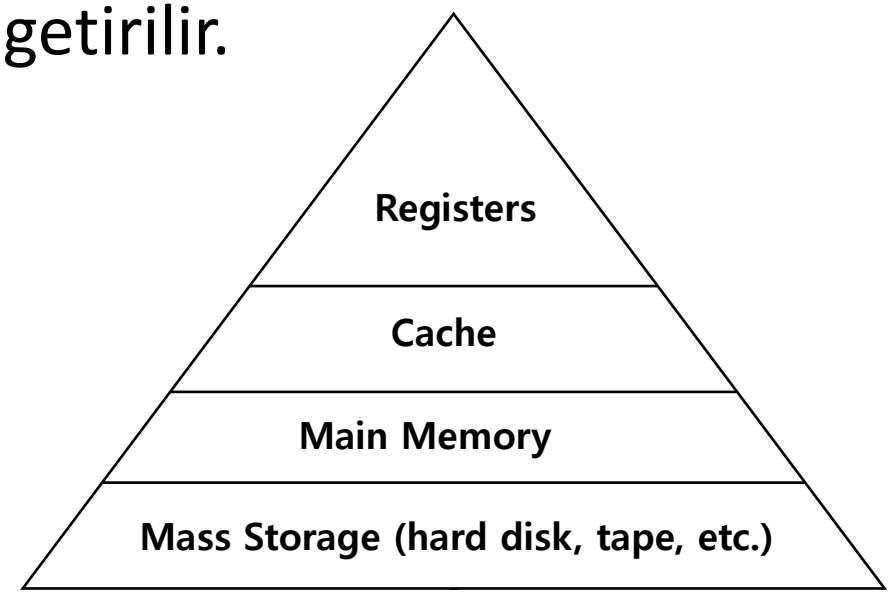
- Sistemin her fiziksel sayfası için bir girdisi olan tek bir sayfa tablosu tutmak.
- Girdi, bize bu sayfayı hangi işlemin kullandığını ve bu işlemin hangi sanal sayfasının bu fiziksel sayfayla eşleştiğini söylemektedir.

21. Fiziksel Belleğin Ötesinde: Mekanizmalar

Operating System: Three Easy Pieces

Fiziksel Belleğin Ötesinde: Mekanizmalar

- Bellek hiyerarşisinde ek bir seviyenin daha kullanımı gerekli olabilir.
- İşletim sisteminin, o an fazla talep görmeyen adres uzayını saklamak için bir yere ihtiyacı vardır.
- Modern sistemlerde, bu rol genellikle bir sabit disk sürücüsü tarafından yerine getirilir.



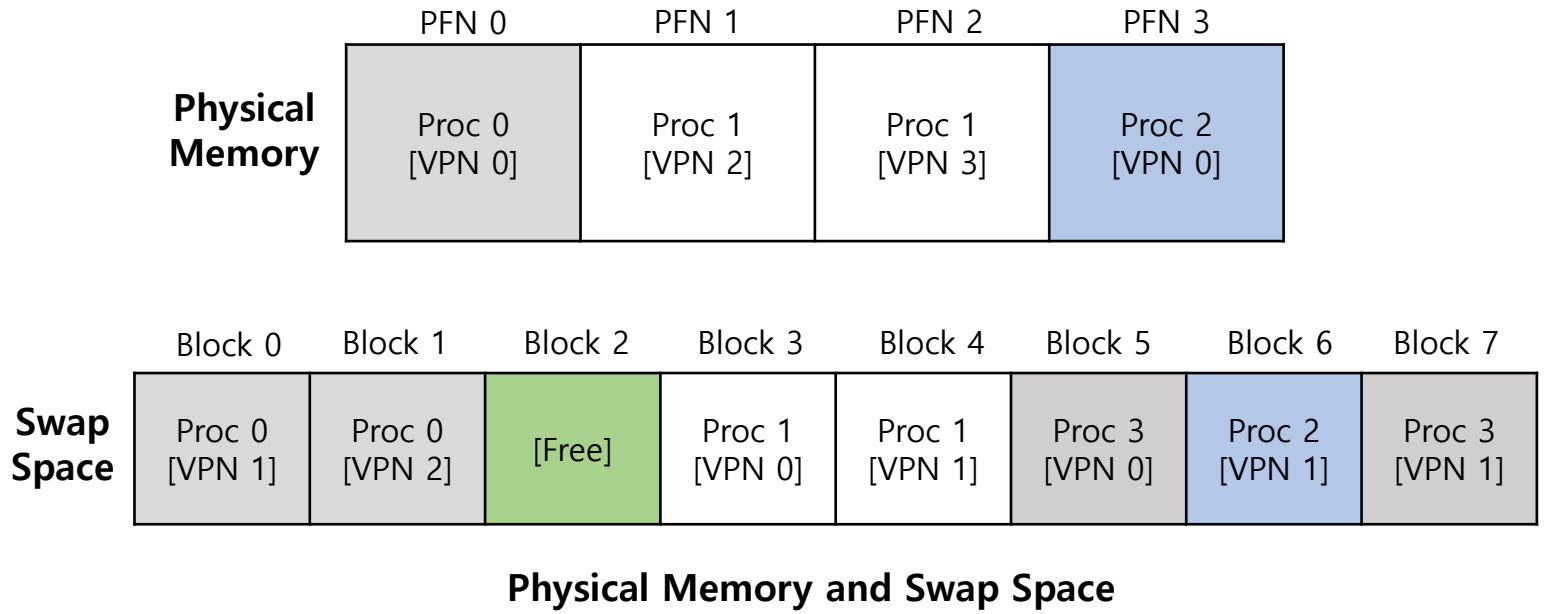
Memory Hierarchy in modern systems

Bir işlem için tek büyük adres uzayı

- Bir fonksiyonu çağırmadan veya verilere erişmeden önce daima kodun veya verilerin bellekte olmasını ayarlamanız gerekir.
- O an bellekte kendisine yer ayrılmış tek bir işlem yok.
- Takas alanının (swap space) eklenmesi, işletim sisteminin aynı anda çalışan birden çok işlem için büyük sanal bellek yanılsamasını desteklemesine olanak tanır.

Takas Alanı (Swap Space)

- Sayfaları gerektiğinde taşımak için diskte biraz yer ayırmak.
- İşletim sisteminin takas alanını hatırlaması gerekir (sayfa boyutlu birim kullanarak)



Physical Memory and Swap Space

Mevcut Biti (Present Bit)

- Sayfaları diske yazmayı ve diskten getirmeyi desteklemek için sistemde donanımsal bazı eklemeler yapılır.
- Bu sayede, Donanım, PTE'ye baktığında, sayfanın fiziksel bellekte bulunmadığını görebilir.

Value	Meaning
1	page is present in physical memory
0	The page is not in memory but rather on disk.

Bellek Zaten Doluysa?

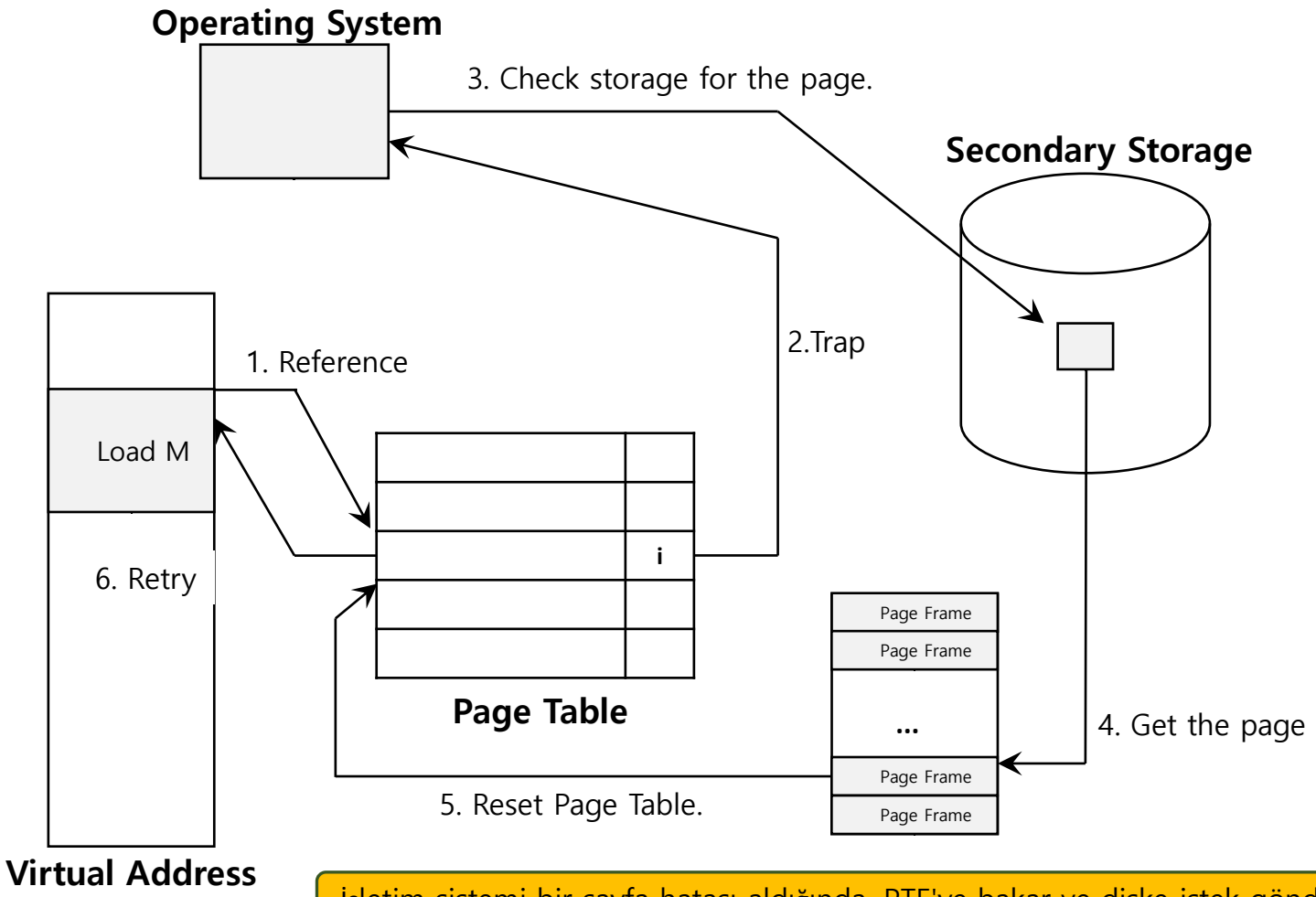
- İşletim sistemi, getirmek üzere olduğu yeni sayfalara yer açmak için bazı sayfaları bellekten çıkartacaktır.
- Çıkarılacak veya değiştirilecek sayfayı seçme süreci sayfa değiştirme politikası (page-replacement policy) olarak bilinir.

Sayfa Hatası (Page Fault)

- Fiziksel bellekte olmayan sayfaya erişilmesi durumu.
- Bir sayfa bellekte mevcut değilse ve diskteyse, işletim sisteminin sayfa hatasını gidermek için sayfayı belleğe getirmesi gerekir.

Sayfa Hatası Kontrol Akışı

- PTE, disk adresi için sayfanın PFN veri alanını kullanabilir.



İşletim sistemi bir sayfa hatası aldığında, PTE'ye bakar ve diske istek gönderir.

Sayfa Hatası Kontrol Akışı – Donanım

```
1:      VPN = (VirtualAddress & VPN_MASK) >> SHIFT
2:      (Success, TlbEntry) = TLB_Lookup(VPN)
3:      if (Success == True) // TLB Hit
4:      if (CanAccess(TlbEntry.ProtectBits) == True)
5:              Offset = VirtualAddress & OFFSET_MASK
6:              PhysAddr = (TlbEntry.PFN << SHIFT) | Offset
7:              Register = AccessMemory(PhysAddr)
8:      else RaiseException(PROTECTION_FAULT)
```

Sayfa Hatası Kontrol Akışı – Donanım

```
9:      else // TLB Miss
10:     PTEAddr = PTBR + (VPN * sizeof(PTE))
11:     PTE = AccessMemory(PTEAddr)
12:     if (PTE.Valid == False)
13:             RaiseException(SEGMENTATION_FAULT)
14:     else
15:     if (CanAccess(PTE.ProtectBits) == False)
16:             RaiseException(PROTECTION_FAULT)
17:     else if (PTE.Present == True)
18:     // assuming hardware-managed TLB
19:             TLB_Insert(VPN, PTE.PFN, PTE.ProtectBits)
20:             RetryInstruction()
21:     else if (PTE.Present == False)
22:             RaiseException(PAGE_FAULT)
```

Sayfa Hatası Kontrol Akışı – Yazılım

```
1:        PFN = FindFreePhysicalPage()
2:        if (PFN == -1) // no free page found
3:                PFN = EvictPage() // run replacement algorithm
4:                DiskRead(PTE.DiskAddr, pfn) // sleep (waiting for I/O)
5:                PTE.present = True // update page table with present
6:                PTE.PFN = PFN // bit and translation (PFN)
7:                RetryInstruction() // retry instruction
```

- The OS must find a physical frame for the soon-be-faulted-in page to reside within.
- If there is no such page, waiting for the replacement algorithm to run and kick some pages out of memory.

Sayfa Değiştirmeleri Ne Zaman Olur?

- İşletim sistemi, bellek tamamen dolana kadar bekler ve ancak o zaman başka bir sayfaya yer açmak için bir sayfayı değiştirirse:
  - Bu biraz gerçekçi değildir ve işletim sisteminin belleğin küçük bir bölümünü proaktif olarak boş tutması için pek çok neden vardır.
- Swap Daemon, Page Daemon
  - Kullanılabilir sayfalar LW değerinden daha az: bellek boşaltmaktan sorumlu bir arka plan işlemi çalıştır.
  - Kullanılabilir sayfalar HW değerine sahip olana kadar sayfaları çıkar.

22. Fiziksel Belleğin Ötesinde: Politikalar

Operating System: Three Easy Pieces

Fiziksel Belleğin Ötesinde: Politikalar

- **Bellek baskısı**, işletim sistemini aktif olarak kullanılan sayfalara yer açmak için bazı sayfaları bellekten çıkarmaya zorlar.
- Hangi sayfanın çıkarılacağı kararı, işletim sisteminin bi bileşeni olan sayfa değiştirme politikası tarafından verilir.

Önbellek Yönetimi

- Bir önbellek (şu anki durumda ana bellek) için bir değiştirme politikasının amacı, önbellek ıska sayısını en aza indirmektir.
- Önbelleğe isabet ve ıska sayıları, ortalama bellek erişim süresini (AMAT) hesaplamamızı sağlar.

$$AMAT = (P_{Hit} * T_M) + (P_{Miss} * T_D)$$

Argument	Meaning
T_M	The cost of accessing memory
T_D	The cost of accessing disk
P_{Hit}	The probability of finding the data item in the cache(a hit)
P_{Miss}	The probability of not finding the data in the cache(a miss)

Optimal Değiştirme Politikası

- Genel olarak en az sayıda ıskalamaya yol açan politika.
- En uzak gelecekte erişilecek olan sayfanın değiştirilmek için seçilmesi.
- Mümkün olan en az ıskayla sonuçlanır.
- Optimal Değiştirme Politikasını gerçekleştirmek mümkün müdür?

Optimal Değiştirme Politikası

- Genel olarak en az sayıda ıskalamaya yol açan politika.
- En uzak gelecekte erişilecek olan sayfanın değiştirilmek için seçilmesi.
- Mümkün olan en az ıskayla sonuçlanır.
- Optimal Değiştirme Politikasını gerçekleştirmek mümkün müdür?
  - Hayır, mükemmele ne kadar yakın olduğumuzu bilmek için yalnızca bir karşılaştırma noktası (benchmark) işlevi görür.

Optimal Politika Örneği

Reference Row

0 1 2 0 1 3 0 3 1 2 1

Access	Hit/Miss?	Evict	Resulting Cache State
0	Miss		0
1	Miss		0,1
2	Miss		0,1,2
0	Hit		0,1,2
1	Hit		0,1,2
3	Miss	2	0,1,3
0	Hit		0,1,3
3	Hit		0,1,3
1	Hit		0,1,3
2	Miss	3	0,1,2
1	Hit		0,1,2

Hit rate is $\frac{Hits}{Hits+Misses} = \mathbf{54.6\%}$

Gelecek bilinemez.

Basit Bir Politika: FIFO

- Sayfalar sisteme girdiklerinde sıraya alınır.
- Bir değiştirme gerçekleştiğinde, sıranın sonundaki sayfa ("İlk giren" sayfa) çıkarılır.
- Uygulaması basittir, ancak sayfa önemi belirlenemez.

Örnek: FIFO Politikası

Reference Row

0 1 2 0 1 3 0 3 1 2 1

Access	Hit/Miss?	Evict	Resulting Cache State
0	Miss		0
1	Miss		0,1
2	Miss		0,1,2
0	Hit		0,1,2
1	Hit		0,1,2
3	Miss	0	1,2,3
0	Miss	1	2,3,0
3	Hit		2,3,0
1	Miss		3,0,1
2	Miss	3	0,1,2
1	Hit		0,1,2

Hit rate is $\frac{Hits}{Hits+Misses} = \mathbf{36.4\%}$

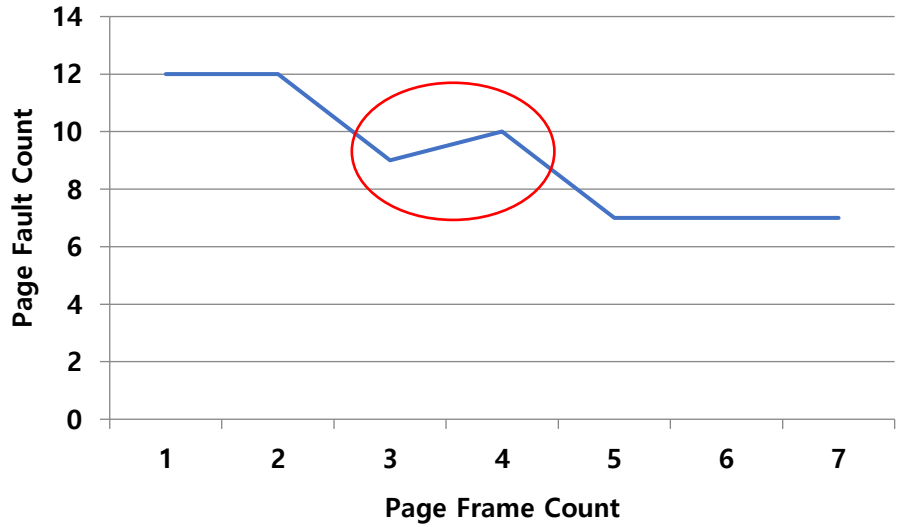
Even though page 0 had been accessed a number of times, FIFO still kicks it out.

BELADY'NİN ANOMALİSİ

- Önbellek büyüdükçe önbellek isabet oranının artmasını bekleriz. Ancak bu örnekte, FIFO ile durum daha da kötüleşebiliyor.

Reference Row

1 2 3 4 1 2 5 1 2 3 4 5



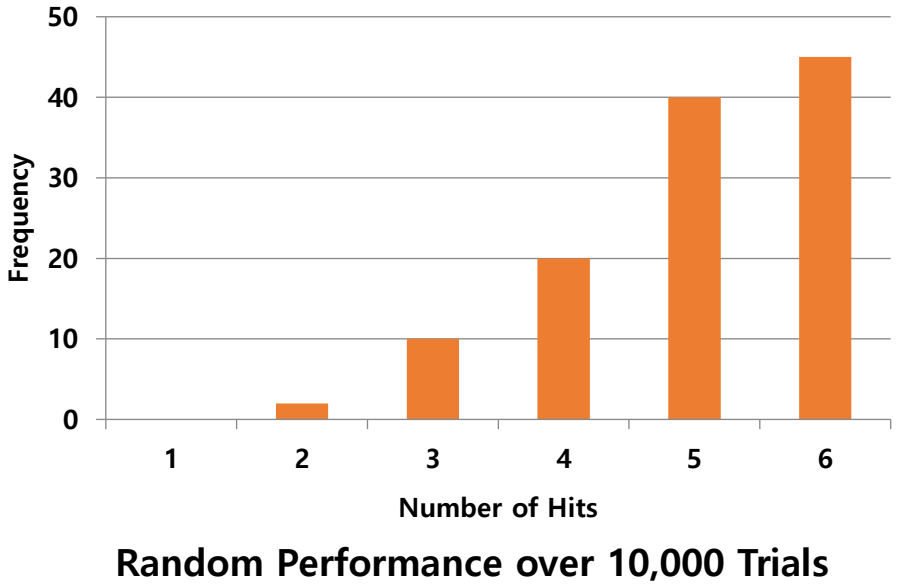
Başka Bir Basit Politika: Rasgele

- Bellek baskısı altında değiştirmek için rastgele bir sayfa seçer.
- Hangi blokların boşaltılacağını seçerken çok akıllı olmaya çalışmaz.
- Performans tamamen rastgele seçimde ne kadar şanslı olduğumuza bağlıdır.

Access	Hit/Miss?	Evict	Resulting Cache State
0	Miss		0
1	Miss		0,1
2	Miss		0,1,2
0	Hit		0,1,2
1	Hit		0,1,2
3	Miss	0	1,2,3
0	Miss	1	2,3,0
3	Hit		2,3,0
1	Miss	3	2,0,1
2	Hit		2,0,1
1	Hit		2,0,1

Rastgelenin Performansı

- Bazen, Rastgele, örnekte de görüldüğü gibi 6 isabet elde ederek optimal kadar iyidir.



Geçmişi Kullanma

- Geçmişten yararlan ve geçmişi kullan.
- İki tür geçmiş bilgisi: (i) yenilik (recency) (ii) sıklık (frequency)

Historical Information	Meaning	Algorithms
recency	The more recently a page has been accessed, the more likely it will be accessed again	LRU
frequency	If a page has been accessed many times, It should not be replaced as it clearly has some value	LFU

Geçmiş Kullanma: LRU

- En önce kullanılmış sayfayı değiştirir.

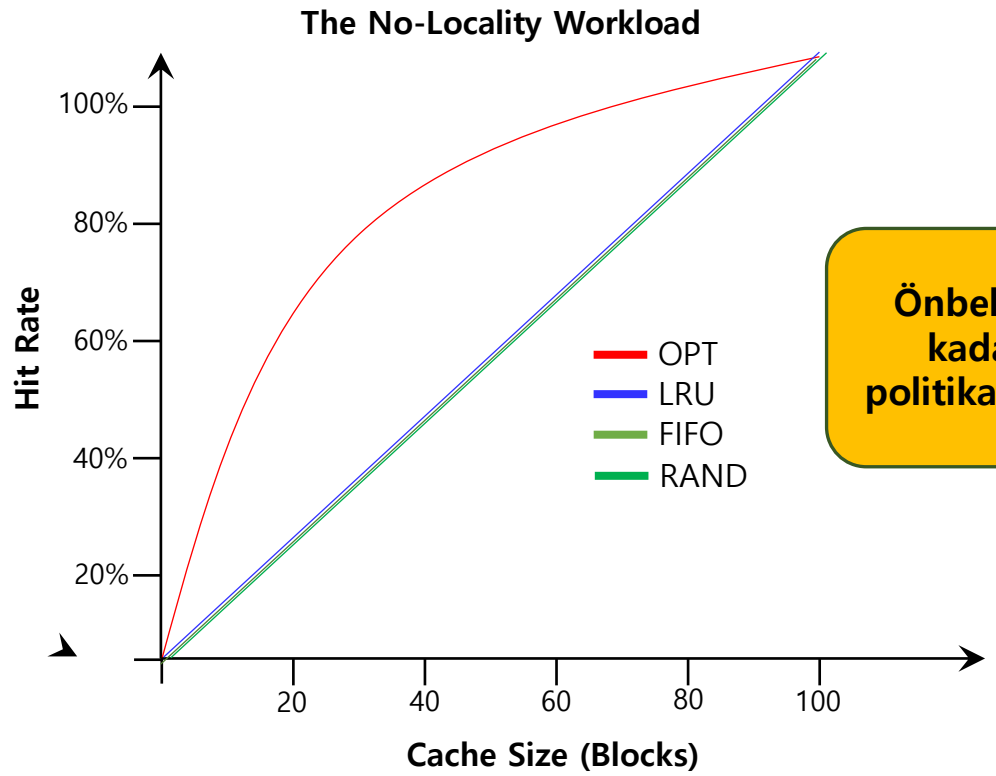
Reference Row

0 1 2 0 1 3 0 3 1 2 1

Access	Hit/Miss?	Evict	Resulting Cache State
0	Miss		0
1	Miss		0,1
2	Miss		0,1,2
0	Hit		1,2,0
1	Hit		2,0,1
3	Miss	2	0,1,3
0	Hit		1,3,0
3	Hit		1,0,3
1	Hit		0,3,1
2	Miss	0	3,1,2
1	Hit		3,2,1

İş Yükü Örneği: Yerel Olmayan İş Yükü

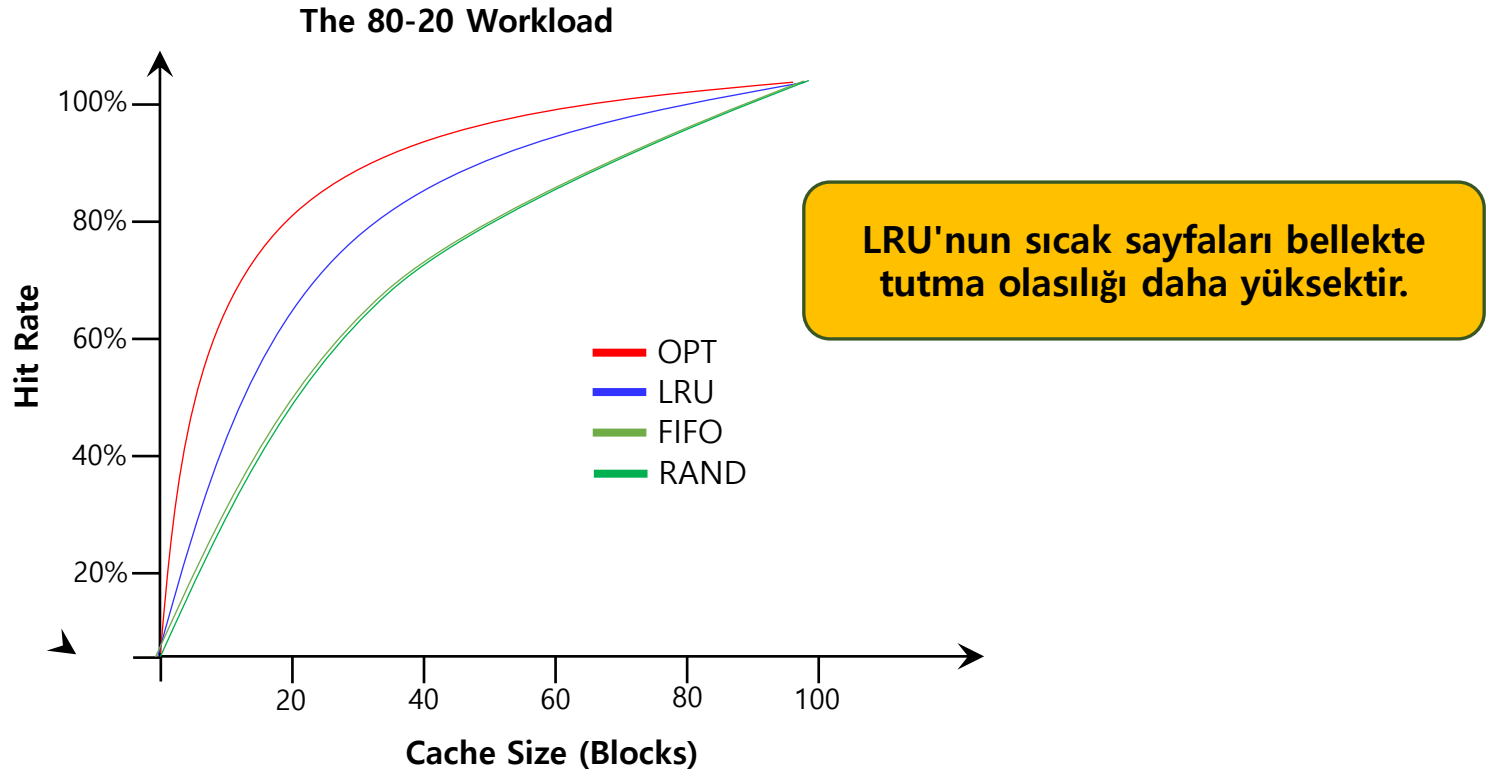
- Her referans, erişilen sayfalar kümesindeki rastgele bir sayfaya yöneliktir.
- İş yükü, zaman içinde 100 benzersiz sayfadan birine rastgele erişir.



Önbellek, tüm iş yükünü sığdıracak kadar büyük olduğunda, hangi politikayı kullandığınız önemli değildir.

İş Yükü Örneği: 80-20 İş Yükü

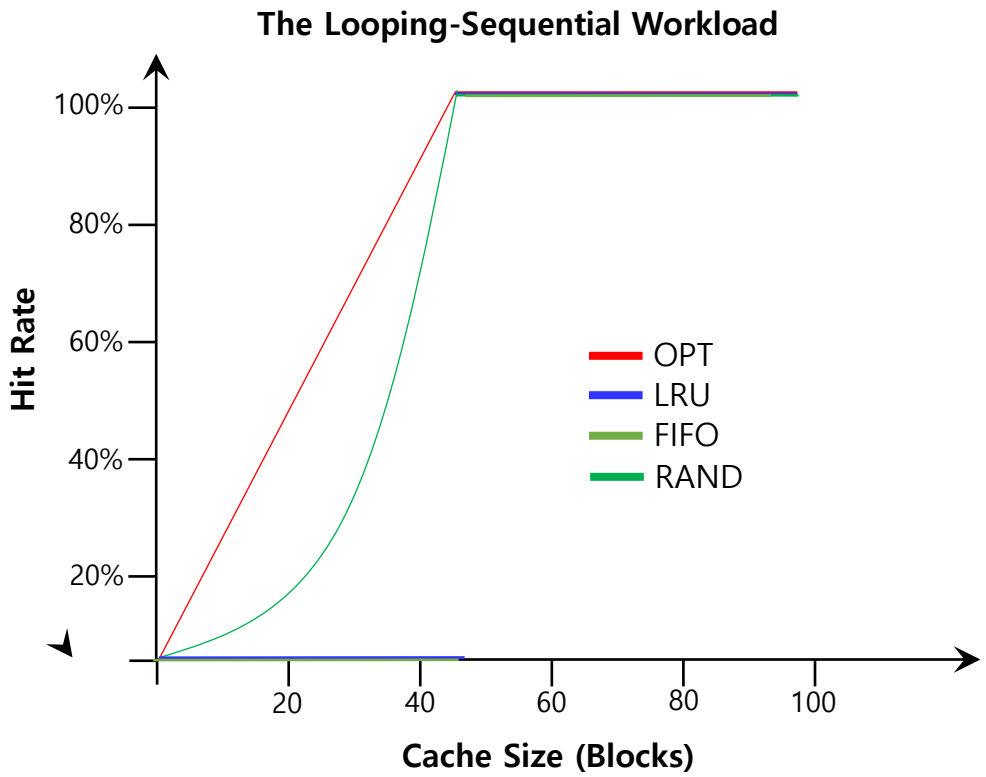
- Yerellik sözkonusu: Referansın %80'i sayfaların %20'sine,
- Kalan %20'lik kısım, sayfaların kalan %80'lik kısmına.



LRU'nun sıcak sayfaları bellekte tutma olasılığı daha yüksektir.

İş Yükü Örneği: Sıralı Döngü

- Sırayla 50 sayfaya referans verilir. 0'dan başlayarak, sonra 1, … 49. sayfaya kadar ve ardından döngü (Toplam 10.000 erişim).



Geçmiş Bilgisi Kullanan Algoritmaların Uygulanması

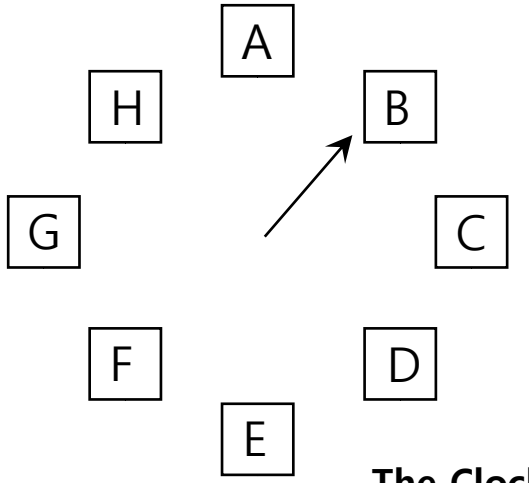
- En az ve en önce hangi sayfaların kullanıldığını takip etmek için, sistemin her bellek referansında bazı hesaplamalar yapması gerekir.
- Bu amaçla donanım desteği eklenmiştir.

LRU’ya Yakınsamak

- Kullanım biti (Use Bit) olarak isimlendirilen donanım desteği vardır.
- Bir sayfaya referans yapıldığında, kullanım biti donanım tarafından 1'e ayarlanır.
- Donanım asla bu biti sıfırlamaz; bu işletim sisteminin sorumluluğundadır.
- Saat Algoritması (Clock algorithm): Sistemin tüm sayfaları dairesel bir liste halinde düzenlenmiştir. Bir saat ibresi, başlamak için belirli bir sayfayı işaret eder.

Saat Algoritması

- Algoritma, 0'a ayarlanmış bir kullanım biti bulana kadar devam eder.



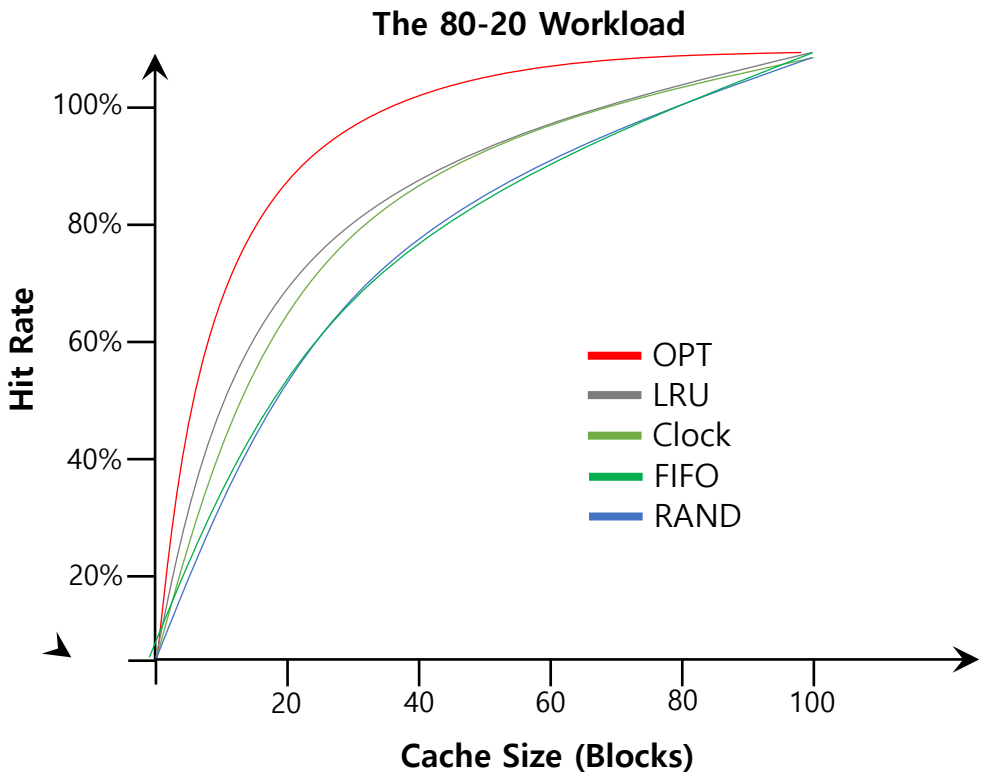
Use bit	Meaning
0	Evict the page
1	Clear **Use bit** and advance hand

The Clock page replacement algorithm

Bir sayfa hatası oluştuğunda, ibrenin işaret ettiği sayfa incelenir.
Alınan eylem Kullanım bitinin değerine bağlıdır.

Saat Algoritması ile İş Yükü

- Saat algoritması, LRU kadar iyi sonuç vermez, geçmişi hiç dikkate almayan yaklaşımlardan ise daha iyidir.



Boşuna Çalışma (Thrashing)

- Belleğe aşırı talep vardır ve çalışan işlemler kümesinin bellek talepleri, kullanılabilir fiziksel belleği aşmaktadır.
- İşlemlerin bir alt kümesini çalıştırmamaya karar vermek gerekir. Azaltılmış işlem kümesi, belleğe sığar.