Sanal Bellekler
(Virtual Memories)

Sanal Bellek (Dolaylı Gösterim)

- Aslında sahip olduğunuzdan daha fazla belleğe ihtiyacınız var ise, ne yapılabilir?
- Sanal bellek (ing: Virtual Memory) çözümdür!
  - Sanal bellek neden lazım onu konuşacağız.
  - Bu bellekler nasıl uygulamaya geçer konuşacağız.
  - Son olarak, sanal bellek nasıl hızlandırılır ona bakacağız: Translation Lookaside Buffers (TLBs).
- Bir sonraki haftaya MIPS diline geçeceğiz.

Gerçek Problemler

- Aslında sahip olduğunuzdan daha fazla belleğe ihtiyacınız var ise, ne yapılabilir?
  - **Programlarınızı diskte tutar, belleği de cache gibi kullanırsanız, bu sanal bellek yapısıdır.**
  - **Sanal bellekten önce, programcılar çalışacak kod ve veri parçalarını yüklemeden önce diski kapatırlardı.**
    - **Sıkıcı ve hataya açık bir yol idi.**

- Aynı anda birden çok program ile çalışmamız gerektiğinde bambaşka problemler doğuyor.
  - Her programınız elinizdeki belleğe ayrı ayrı sığsa bile, çalışan 10 program sığmayacaktır.
  - Birden çok program aynı bellek adresine yazmak isteyebilir.
  - Bir programın verilerini bir başka program okuyup yazamasın diye nasıl koruyacağız?
    - **Sonraki slayta bakınız.**

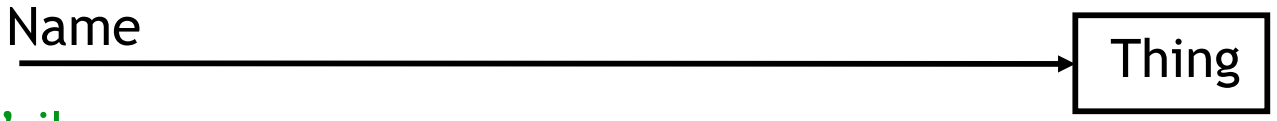
Gerçek Problemler (devamı)

- Aynı anda birden çok program ile çalışmamız gerektiğinde bambaşka problemler doğuyor.
  - Her programınız elinizdeki belleğe ayrı ayrı sığsa bile, çalışan 10 program sığmayacaktır.
    - **Tamamen önceki slayttaki problemin aynısıdır.**
  - Birden çok program aynı bellek adresine yazmak isteyebilir.
    - **Yani, Program A ve B yığıt (ing: stack) bellek başlangıç adreslerini 0x10000000 olarak benzer belirleyebilirler.**
    - **Her programı, birbirinden ayrı adres seti kullanacak şekilde, bir de farklı farklı adres setlerine göre derlemek pratik değildir, uygulamada imkansızdır.**
  - Bir programın verilerini bir başka program okuyup yazamasın diye nasıl koruyacağız?

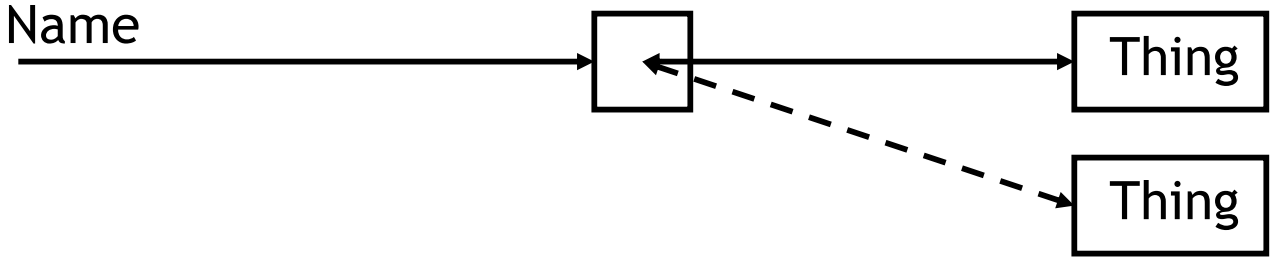
Dolaylı Gösterim (Indirection)

- "Bilgisayar bilimlerindeki her problem bir dolaylı gösterim seviyesi ekleyerek te çözülebilir"
- **Indirection**: Bir şeyin değerini kullanmaktansa bir isim, referans veya kapsayan obje kullanarak ona referans etmeye denilir. Bir isim ve bir nesne arasında ismin kapsadığı yerden bağımsız esnek bir atamadır.

- 'Indirection' olmadan



- 'Indirection' ile

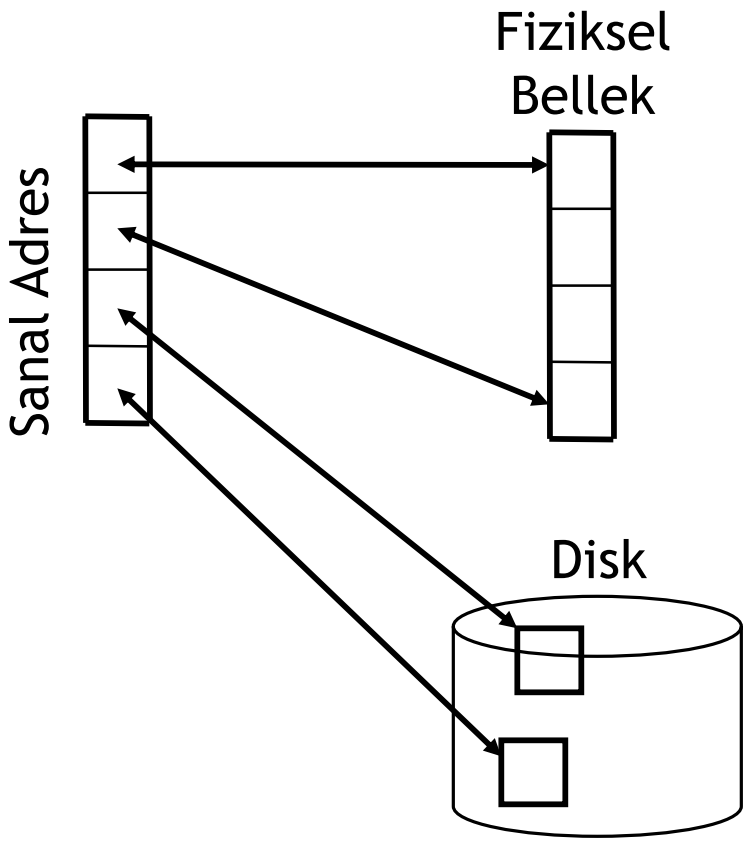


- **Örnekler**:

  Pointer'lar, Domain Name Service (DNS) ismi->IP adresi, snail mail (mail yönlendirme), DHCP, renk paletleri, call centre'lar.

Sanal Bellek

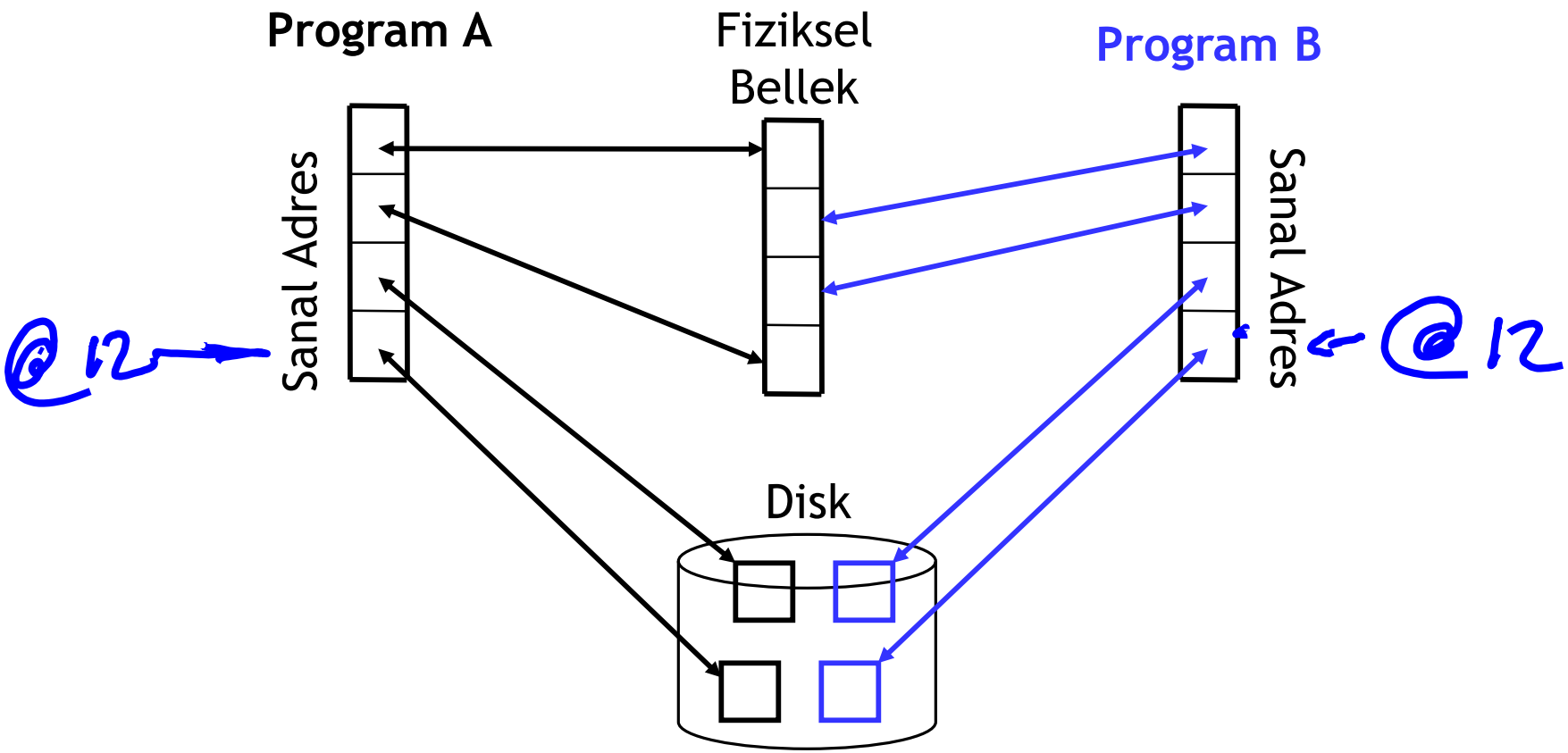
- Programlar kullanarak “sanal adresleri”, makinenin “fiziksel belleği” içindeki yerleri temsil eden “fiziksel adreslere” dönüştürebiliriz.
  - “Dönüştürme” bir dolaylı gösterim/indirection seviyesini ifade eder.



Bir sanal adres fiziksel belleğe veya diske atanabilir.

Sanal Bellek

- Çalışan farklı süreçlerin sanaldan fiziksele farklı atamaları olmasından dolayı, iki program özgürce aynı sanal adresi kullanabilir.
- Fiziksel belleğin A ve B'ye farklı yerleri atanarak, birbirlerinin verilerini okuma/yazma sorunu da ortadan kaldırılır.

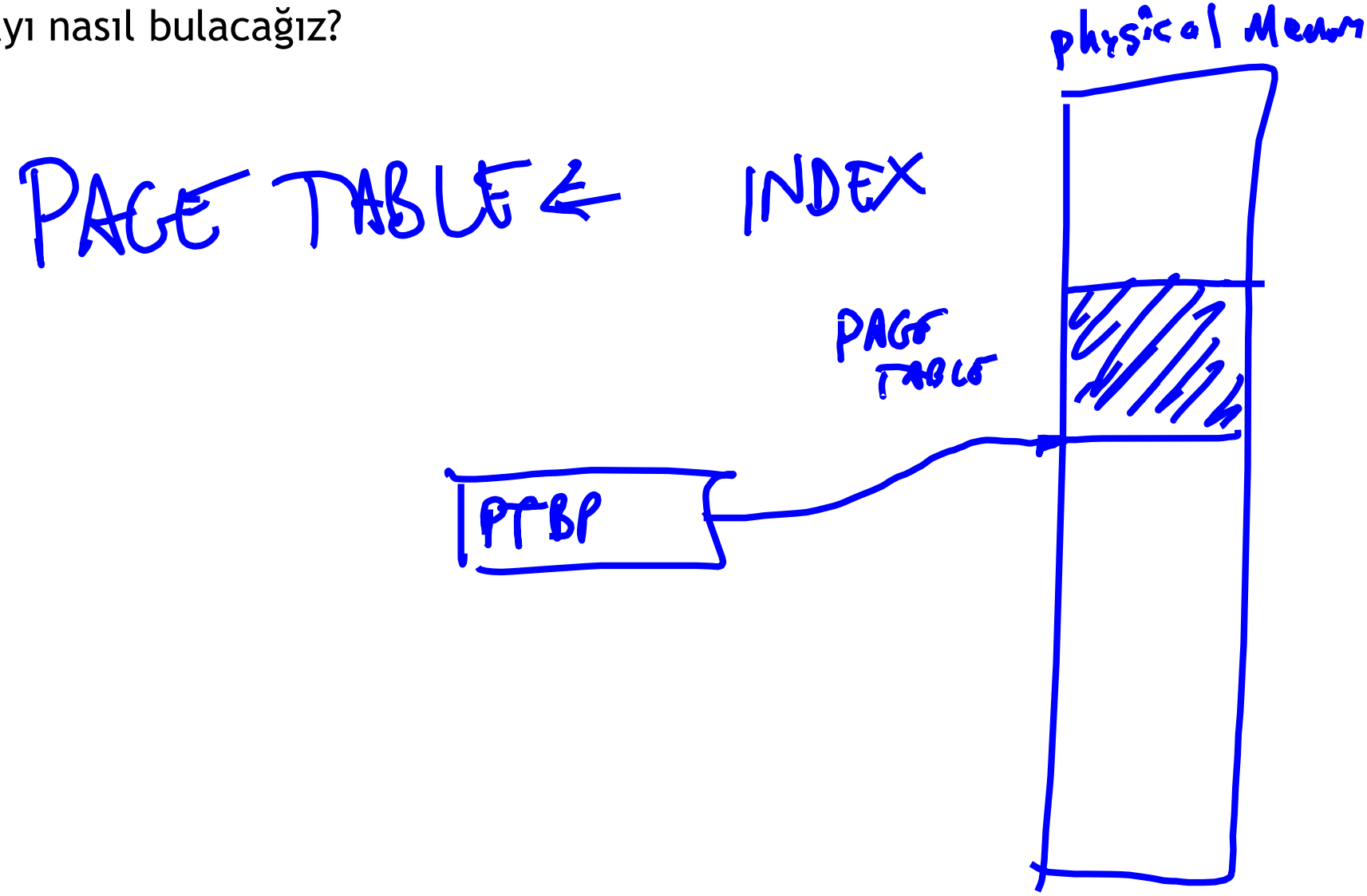


Cache Konusu Hatırlatma

- Bir sistemde adres çevrimi yapılacağı söylendiğinde, o da cache konusuna girmektedir.
  - Diskin büyük alanını ve belleğin performansını kullanmak istiyoruz.
- Sanal bellek sistemleri tasarımı da fiziksel diskin aşırı yavaş ulaşım maliyeti nedeniyle ortaya çıkmıştır.
  - Bellek gecikmesi cache'den yaklaşık **100** kez yavaşken, disk gecikmesi cache'den yaklaşık **100.000** kez, bellekten ise **1.000** kez yavaştır.
    - Yani yavaş olana başvurmak (ing: miss penalty) gerçek bir sorundur.
- Biz yine miss oranını düşürmeyi hedefliyoruz:
  - Sanal bellek "sayfaları" (**1KB→8K**) cache bloklarından çok daha büyüktür. Niye? (yine **uzaysal yerellik**)
  - Burada rahatça 'fully associative' cache yapılabilir.
    - En az kullanılan satırı da yaklaşımla buluruz.
- Peki yazmada 'write-through' nu, 'write-back' mi kullanılır?

Doğru Sayfayı Bulmak

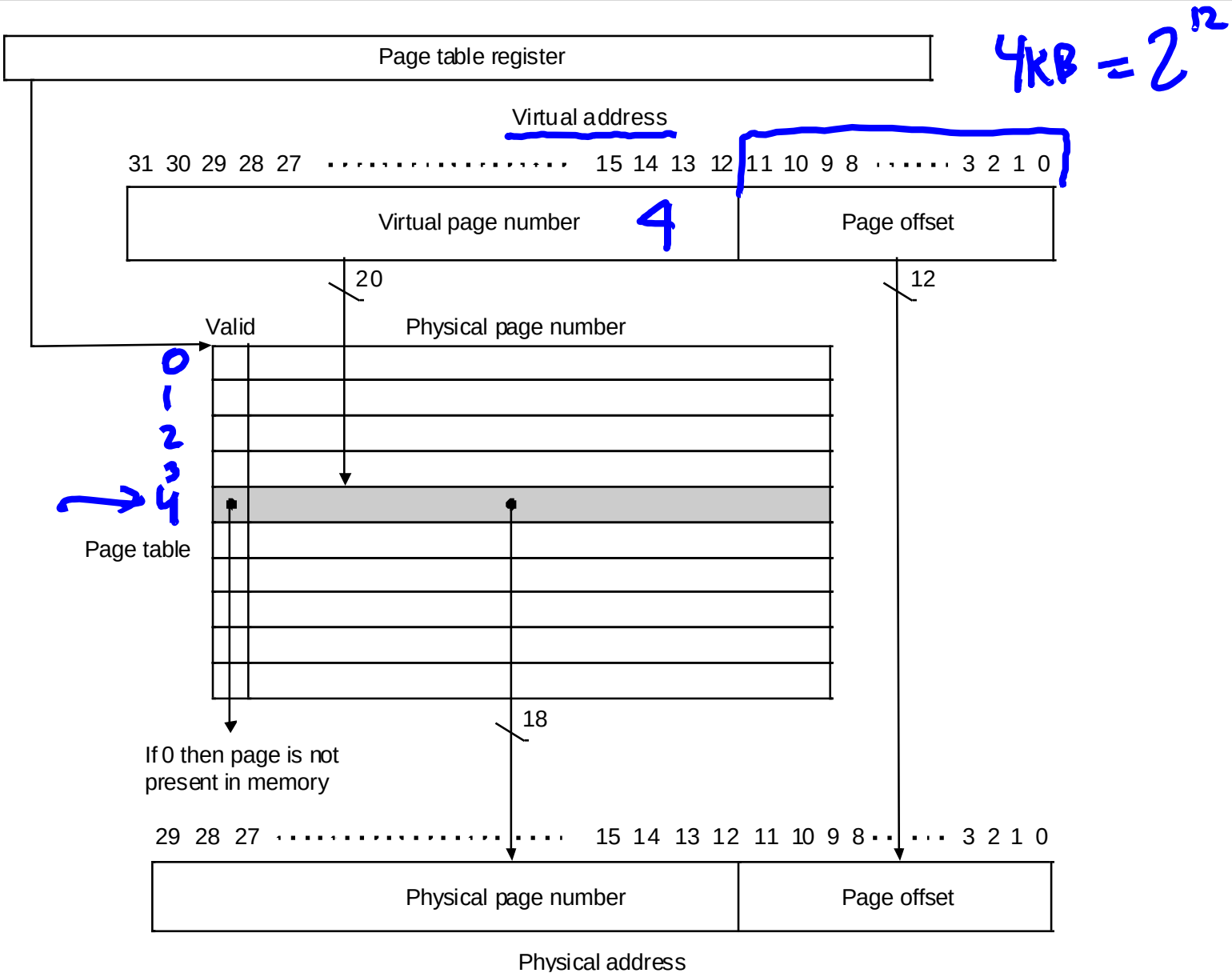
- Eğer ‘fully associative’ kullanacaksak, tüm belleği taramadan doğru sayfayı nasıl bulacağız?



Doğru Sayfayı Bulmak

- Eğer ‘fully associative’ kullanacaksak, tüm belleği taramadan doğru sayfayı nasıl bulacağız?
  - Kitaplardaki indeks gibi bir yapı kullanacağız.
- Buradaki indeksleme işine sayfa tablosu (ing: **page table**) diyoruz.
  - Her sürecin ayrı bir sayfa tablosu vardır.
    - Bir sayfa tablosu kaydı (ing: page table register) şu anki sürecin sayfa tablosunun yerini gösterir (bir pointer).
  - Sayfa tablosu **sanal sayfa numarası** (ing: virtual page number, (VPN)) ile indekslenir.
    - VPN numarası, sayfa ofseti hariç diğer tüm bitlerden oluşur.
  - Her kayıtta bir ‘valid bit’ ve **bir fiziksel sayfa numarası (ing: physical page number** (PPN)) vardır.
    - PPN ile sayfa ofset birleştirilirse fiziksel adres elde edilir.
  - Tag bilgisine gerek yok indeks, VPN’in tamamından oluşur..

Sayfa Tablosu Şeması



Sayfa Tablosu Ne Kadar Büyür?

- Önceki slayta göre:
  - Sanal sayfa numarası 20 bittir.
  - Fiziksel sayfa numarası 18 bit + valid bit -> 32 bit yapısında.

19

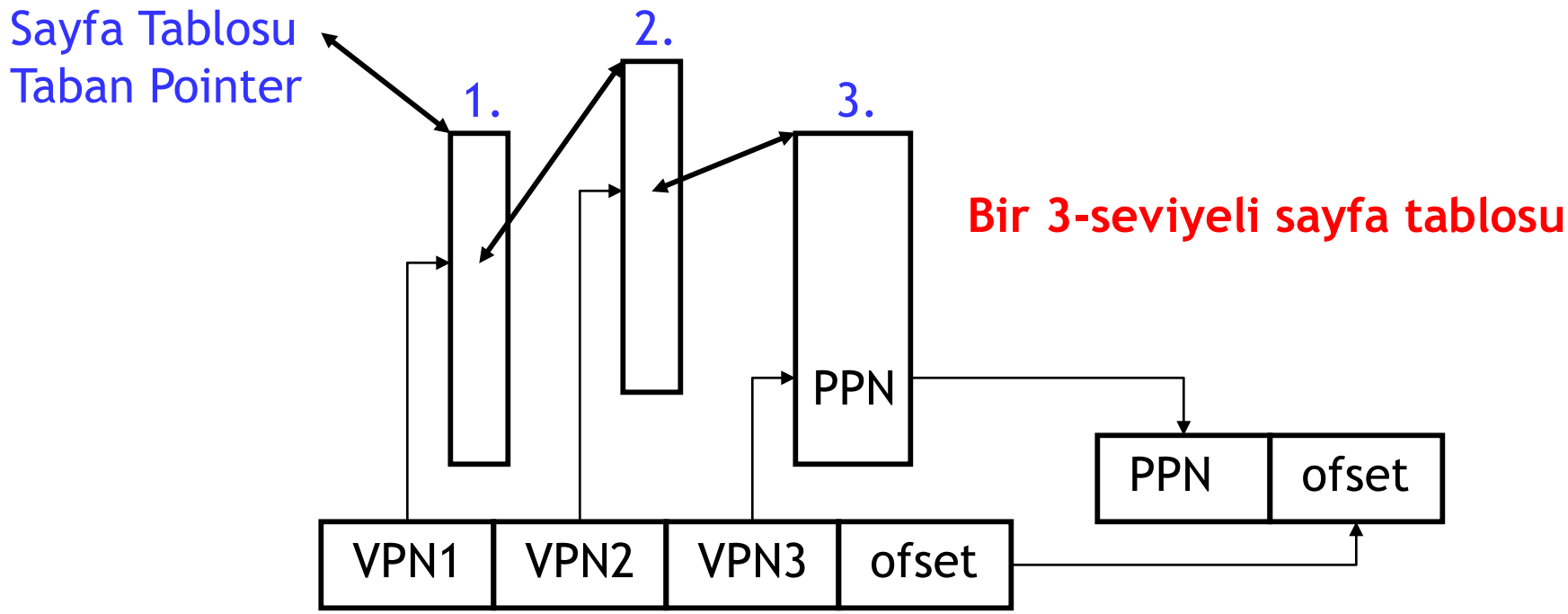
2^{20} = 1 million × 4B = 4MB

- 64 bit mimaride rakam uçuyor mu? Sonraki slaytta.

2^{52}

Büyük Sayfa Tabloları ile Baş Etmek

- Çok seviyeli sayfa tabloları kullanılır. Ne demiştik:
  - "Bilgisayar bilimlerindeki her problem bir dolaylı gösterim seviyesi ekleyerek te çözülebilir" veya iki seviye...



- 64 bitteyiz diye çoğu sürecin 2^{64} gibi bir adres kullanımı yok ki...Gerekli olmayan seviyeler o yüzden kullanılmaz. Hatta 2. ve 3. seviyeyeler diske yazılabilir.

Ama Bir Saniye!

- Seviyelerde her MEM[adres] bellek erişimini şöyle yaparız:

  MEM[MEM[MEM[MEM[PTBR + VPN1<<2] + VPN2<<2] + VPN3<<2] + ofset]

  – *Yani;* 4 bellek erişimi

- Ve henüz **en kötü senaryomuzu konuşmadık** (*yani*; 'page fault')…

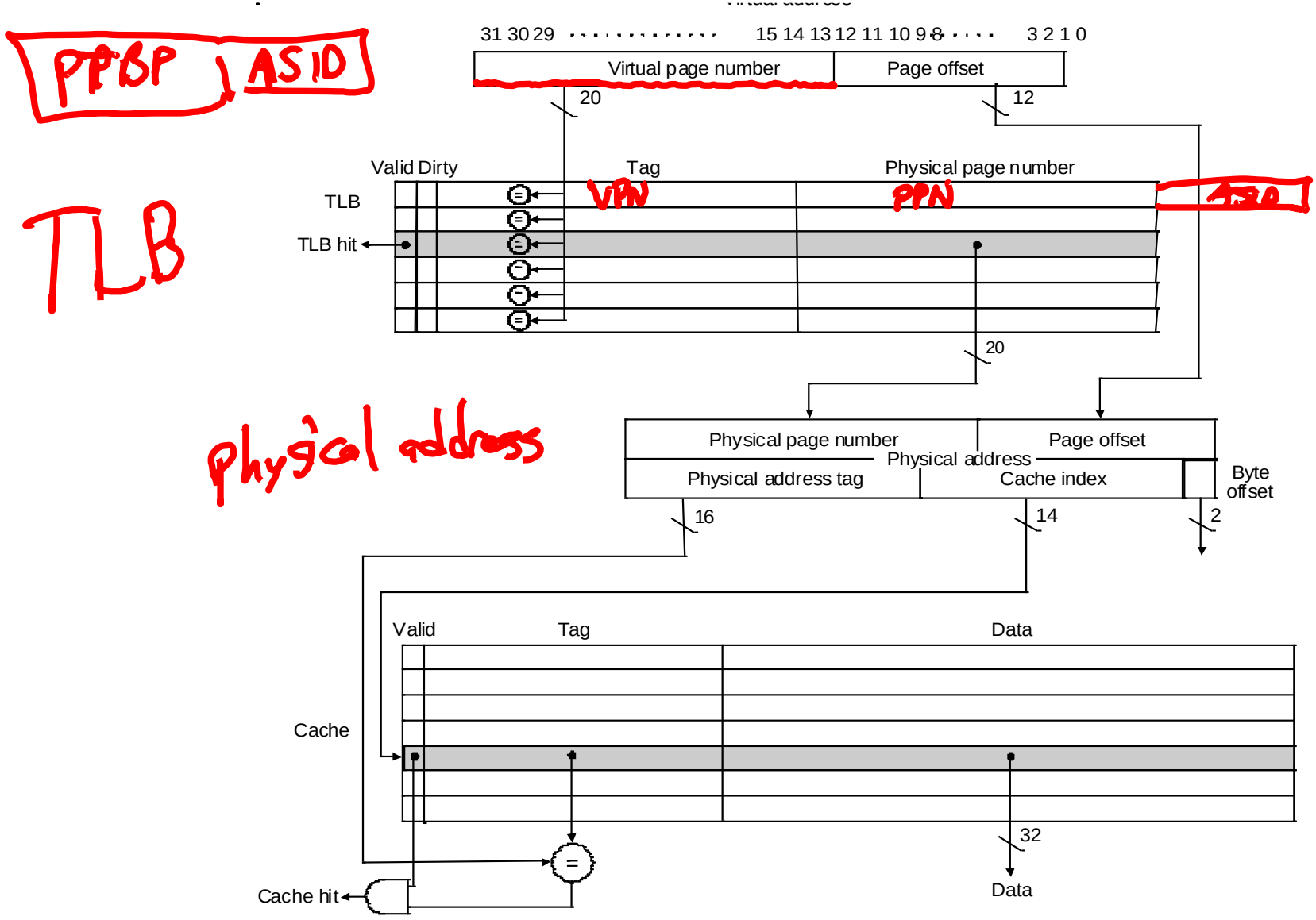
"Bilgisayar bilimlerindeki her problem bir dolaylı gösterim seviyesi ekleyerek te çözülebilir"

– **Ama çok ta seviyeli olmasın…**

- Peki bu kadar seviyeyle nasıl baş edeceğiz?
  – **Cache** kullanmak gerek!

Dönüşümü Cache'lemek

- Sanal → Fiziksel dönüşüm **Translation Lookaside Buffer** (TLB) isimli işlemci içi bir cache'de tutulur.



Peki Ya TLB Miss?

- TLB'de miss alırsak, "sayfa tablosunda gezinmemiz" lazımdır.
  - MIPS/RISC/ARM'de bir hata tetiklenir ve yazılımla TLB doldurulur
    - MIPS'te "TLB_write" komutları vardır.
  - Ix86'da da bir "hardware page table walker" TLB'yi doldurur

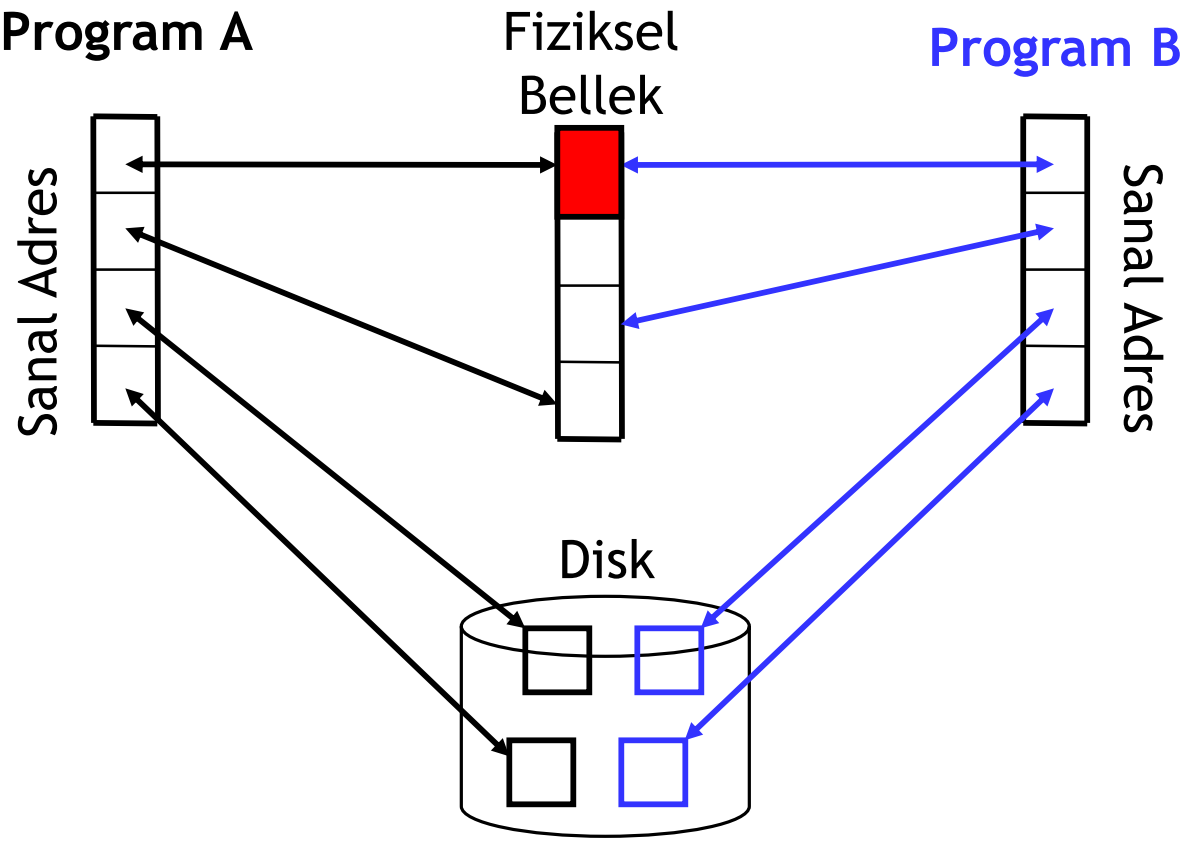
- TLB'de olup, bellekte o sayfayı bulamazsak ne olur?
  - Bu durum sayfa hatasıdır (ing: **page fault**)
  - İşletim sistemi diskten sayfayı talep eder.
  - Üzerine yazacak bir sayfa seçmeye ihtiyaç duyar.
    - İşletim sistemi (O/S) yaklaşık en az kullanılan yöntemini dener (ing: approximate LRU).
  - Üzerine yazılan sayfa 'dirty' ise geri diske yazılmalıdır.

Bellek Koruması

- Bellekte çalışan süreçlerin birbirlerinin verilerini okuyup/yazmamaları için bir sürecin kendi Sanal→Fiziksel dönüşümlerini değiştiremeyeceğinden emin olmamız gerekir.
- Tipik olarak bunu yapmanın yolu:
  - İki işlemci moduyla çalışılır: user ve kernel (programlar ve O/S).
    - Yalnızca O/S kernel moda çalışır.
  - Sadece kernel modda sanal belleğe yazmaya izin verilir. Bu modda değiştirilenler:
    - Sayfa tablosu/The page table
    - Sayfa tablosu taban pointeri/The page table base pointer
    - TLB

Belleği Paylaşmak

- Sayfa sayfa yapılmış sanal bellek aynı zamanda iki sayfa tablosu da aynı fiziksel adrese işaret ederse bellek paylaşımına izin verir.
- Örneğin bir programın iki kopyasını çalıştırdığınızda, O/S programlar arasında kod sayfalarını paylaştırır.



Özet

- Sanal Bellek kesin cennetten gelmedir:
  - Bizi bellek işlemleri yapmaktan kurtarır.
  - Farklı programlarda aynı sanal adreslerin kullanılmasına müsaade eder.
  - Farklı süreçleri birbirlerinden korur.
  - Süreçler arasında kontrollü bir paylaşım imkanı sağlar (gerçi pek esnek de değil).
- Anahtar teknik dolaylı gösterim/**indirection**:
  - Bilgisayar bilimlerindeki her problem bir dolaylı gösterim seviyesi ekleyerek te çözülebilir.
- Cache mantığıyla yeni imkanlar ortaya çıkar:
  - Sanal bellek, fiziksel belleğin disk için bir cache olarak kullanılmasını sağlar.
  - Burada cache'i (Translation Lookaside Buffer) Sanal Belleğin daha hızlı dolaylı gösterim yapması için kullanırız.