Bellek Hiyerarşisi

Bellek Teknolojileri

Rastgele Erişim Bellekler RAM (Random Access Memory):

- Günümüzde kullanılan tüm yarıiletken bellekler rastgele erişimlidir (sırasal değil)

Ram özellikleri:

- İşlemciler bellek gözlerine hem okumak hem de yazmak için doğrudan ve hızla erişebilirler
- Bu işlemler elektrik sinyalleri ile gerçekleştirilir
- Bu tip bellekleri read/write bellek olarak adlandırmak daha doğru olur
- RAM uçucu bellektir (volatile) güç kesilirse veriler kaybolur
- Bu nedenle ram sadece geçici (temporary) saklama birimi olarak kullanılır

Bellek Hiyerarşisi

Bellek Teknolojileri

Belleklerde Gecikme Ölçüleri:

Erişim Süresi (Access time): Erişim isteğinin başlaması ile adreslenen sözcüğün okunması/yazılması arasında geçen süre

Çevrim Süresi (Cycle time): İki bellek erişimi arasındaki olası en kısa süre

İki tür RAM vardır:

Dinamik RAM DRAM (Dynamic RAM)

Static RAM SRAM (Static RAM)

Bellek Hiyerarşisi

Bellek Teknolojileri

DRAM (Dynamic RAM):

- Ana belleklerde kullanılır
- DRAM hücreleri verileri kapasite yükleri şeklinde saklar
- Kapasiteler doğal yapıları nedeniyle yüklerini kaybeder. Bu nedenle dinamik bellekleri periyodik olarak tazelemek gerekir (~8ms)
- Dinamik olarak adlandırılır çünkü güç kesilmese bile depoladığı yük zamanla boşalır
- Tazeleme sırasında bellek kullanılamaz (gecikme)
- Her okumadan sonra veri tekrar yazılır çünkü okuma yükü boşaltır
- Çevrim süresi > Erişim süresi
- Ucuz ve yoğundur

Bellek Hiyerarşisi

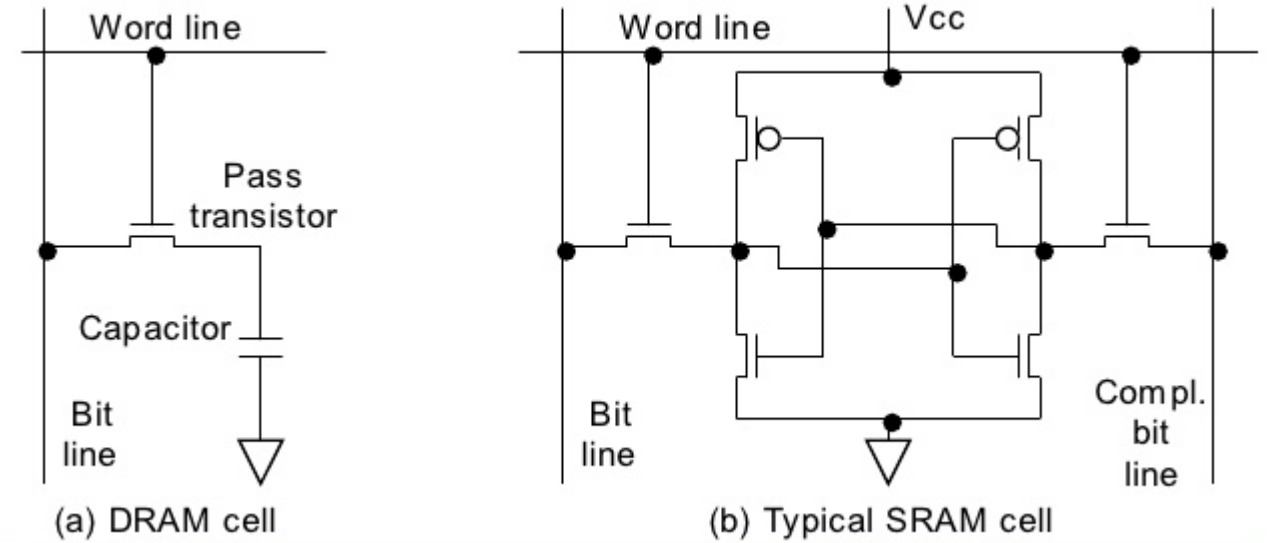
Bellek Teknolojileri

DRAM Teknolojisinde İlerlemeler:

- ✓ SDRAM (Synchronous DRAM):
  - Saat işareti eklenmiştir.
  - Toplu veri aktarımı yapar.
- ✓ DDRAM (Double Data Rate DRAM):
  - Saat sinyalinin hem çıkan hem de inen kenarında erişim yapılır.

Bellek Hiyerarşisi

Bellek Teknolojileri



Bellek Hiyerarşisi

Bellek Teknolojileri

SRAM (Static RAM):

- Cep (Cache) belleklerde kullanılır
- Verileri saklamak için flip-floplar kullanılır
- Bir bit için 6 transistör kullanılır
- Tazeleme gerekli değildir
- Erişim süresi = Çevrim süresi

Bellek Hiyerarşisi

Bellek Teknolojileri

DRAM-SRAM Karşılaştırması:

- İkisi de uçucudur güç sürekli sağlanmalı
- DRAM daha yoğun ve daha ucuzdur
- DRAM tazeleme devresine gerek duyar
- DRAM gecikmesi daha fazladır
- DRAM ucuz SRAM hızlıdır

Bellek Hiyerarşisi

Bellek Hiyerarşisi

Bellek hiyerarşisi çeşitli veri depolama birimlerinin veri iletim hızı/işlem gücüne göre hiyerarşik olarak sıralanmasına verilen addır.

- İşlemcilerin hızı ve işlem gücü arttıkça bilgisayar mimarisini oluşturan veri saklama birimlerinin işlemciyle arasındaki senkron farkı gitgide artmaktadır.
- Bunu aşmak için bir bilgisayarın bellek sistemi, hiyerarşik olarak, seviyelerle düzenlemiştir.
  - Her seviye bir öncekinden daha büyük bir kapasiteye sahip ve daha yavaştır.
- Bellek hiyerarşisine sahip olmanın temel argümanı ekonomidir:
  - CPU hızında çalışan eşsiz büyüklükte bir bellek, mümkün olsa bile oldukça pahalı olacaktır.

Bellek hiyerarşisi fikrini işe yaratan şey yerleşim ilkesidir (principle of locality).

Bellek Hiyerarşisi

Yerleşim İlkesi (The Principle of Locality)

Bir program çalıştırılırken belleğe iki nedenden dolayı erişilir:

- ✓ Komutları okumak
- ✓ Veriyi okumak/yazmak

Belleğe erişim düzgün dağılımlı bir şekilde gerçekleşmez

- ✓ Bazı bölgelerdeki adreslere diğerlerinden daha sık erişilir
- ✓ Bazı adreslere mevcut erişimden kısa bir süre sonra tekrar erişilir
- ✓ Başka bir deyişle, programlar adres alanının bir kısmını tercih etme eğilimindedir.

Bellek Hiyerarşisi

Yerleşim İlkesi (The Principle of Locality)

Zamansal yerleşim (temporal locality):

Belirli bir adrese erişim kısa bir süre sonra tekrar edilme eğilimindedir.

Mekansal yerleşim (spatial locality):

Belirli bir adrese erişim yakındaki adreslere erişim tarafından takip edilme eğilimindedir

- Bu ilkenin sayısal ifadesi 90/10 genel kuralına göre verilmiştir:
- Bir programın çalışma süresinin %90'ı, programın adres alanının %10'una erişmek için harcanır.
- Bu durumda bellek için hiyerarşik bir sistem düşünmek oldukça makuldür
- Bir şekilde en çok kullanılan adresler, adres alanının kabaca %10'unu temsil etmesi gereken hızlı bir belleğe eşlenir ve program çoğu zaman (%90) bu hızlı belleği kullanarak çalışır.
- Belleğin geri kalanı daha yavaş olabilir, çünkü daha az erişilir, ve daha yavaş bellekler daha ucuzdur.

Bellek Hiyerarşisi

Yerleşim İlkesi (The Principle of Locality)

- Bir bellek hiyerarşisinin birkaç seviyesi vardır:
  - En üst seviye
    - CPU'ya en yakın
    - En hızlı (işlemcinin hızına uygun)
    - En küçüktür
- Hiyerarşinin en altına indiğimizde, her seviye bir öncekinden daha yavaş ve daha büyür ama bit başına maliyeti daha düşüktür.
- Hiyerarşinin farklı seviyelerinde tutulan datalar hemen altındaki seviyede de bulunabilir.
- Her seviye altındaki seviyenin bir alt kümesidir.
- Bellek öğeleri (komut veya veri temsil edebilirler) ilk kez çağrıldıklarında daha üst seviyeye getirilirler, çünkü kısa süre sonra tekrar erişilmeleri için iyi bir olasılık vardır.
- Yeni gelecek öğelere yer açılması gerektiğinde ise daha düşük seviyeye geri taşınırlar.

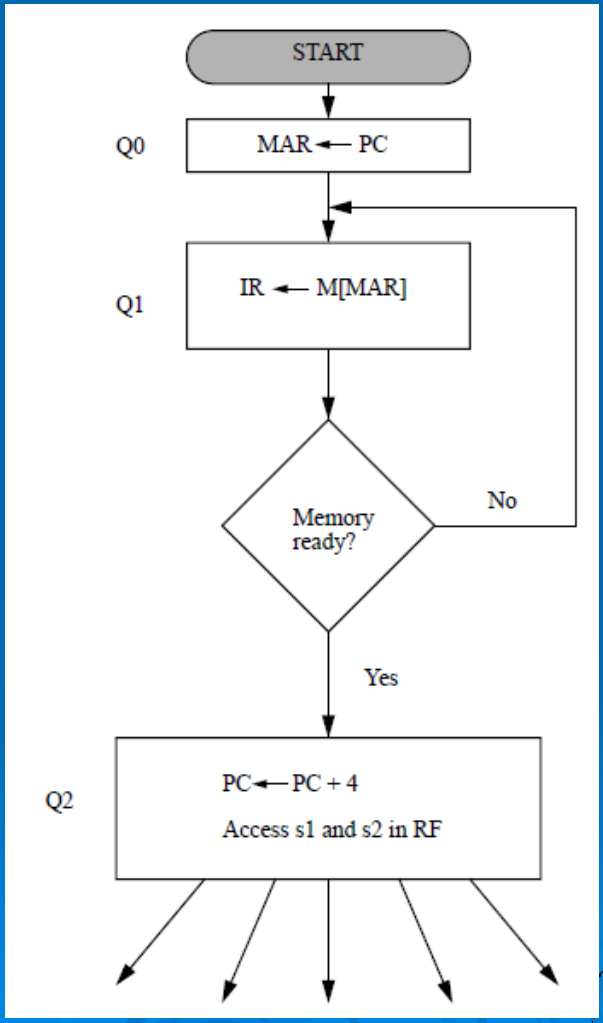
Bellek Hiyerarşisi

Bellek Gecikmesi ve Performansı

- Daha önceki bölümlerde, komut setindeki komutların ideal CPI'i hakkında tartışmıştık.
- O anda belleğin bekleme durumuna geçmeden erişilen öğeye ulaşması için yeterince hızlı olduğunu varsaydık.
- Komut yürütmenin durum diyagramına bakacak olursak, Q1 durumunda MemoryReady sinyalinin bellek döngüsünün tamamlanıp tamamlanmadığını belirlemek için test edildiğini görürüz; eğer değilse, o zaman Kontrol Ünitesi Q1 durumuna geri döner ve hafıza erişimi için gerekli kontrol hatlarını belirtmeye devam eder
- MemoryReady = No olan her saat döngüsü, bu komut için CPI'yı bir artırır
- Aynısı, load veya store komutları için de geçerlidir.
- Belleğin tepki süresi arttıkça, bu komut için gerçek CPI yükselir.

Bellek Hiyerarşisi

Bellek Gecikmesi ve Performansı



Bellek Hiyerarşisi

Bellek Gecikmesi ve Performansı

Bir komutun n ile ifade edilen ideal bir CPI değerine sahip olduğunu varsayalım

Ve komut için k tane adresleme cycle'ı olsun.

- Load / Store komutları için k = 2
- Komut setimizdeki diğer tüm komutlar için k = 1

Bir komut için ideal CPI:

CPI_{ideal} = n (clock cycles per instruction)

Her adresleme döngüsü w tane bekleme saat döngüsü getirirse, bu komut için gerçek CPI

CPI_{real} = n + k * w (clock cycles per instruction)

Bellek Hiyerarşisi

Bellek Gecikmesi ve Performansı

Örnek:

Örnek komut setimiz için ideal CPI'ımızın 4 olduğunu farz ederek gerçek CPI'ı hesaplamak istiyoruz. Komut setimizde Load / Store komutları için 2 tane diğer tüm komutlar için 1 tane adresleme cycle'ı bulunmaktadır. Her bellek erişiminde 1 bekleme çevrimi varsa, gerçek CPI nedir? (Load ve store komutları yürütülen tüm komutların %25'ini oluşturmaktadır)

Cevap:

n = 4

w = 1 (bekleme saat döngüsü)

k = 1 in f1 = 75% (diğer komutlar)

k = 2 in f2 = 25% (load ve store)

$CPI_{real} = 4 + (f1 * 1 + f2 * 2) * w$

$CPI_{real} = 4 + (0.75 * 1 + 0.25 * 2) * 1$

$CPI_{real} = 4 + 1.25 = 5.25$

Bellek Hiyerarşisi

Bazı Tanımlamalar

- Bellek hiyerarşisinde veriler hiyerarşik seviyeler arasında taşınırlar.
- Hiyerarşide seviye ne kadar yüksek olursa, kapasitesi o kadar düşük olur ve mantıksal adres alanının sadece küçük bir bölümünü barındırabilir.
- Seviyeler arasındaki transferler bloklarla gerçekleşir ve bir blok
  - ✓ sabit boyutlu
  - ✓ veya değişken boyutlu olabilir
- Sabit boyutlu bloklar daha çok kullanılırlar ve bu durumda bellek kapasitesi blok boyutunun katı şeklindedir.
- Farklı bellek seviyeleri arasındaki blokların hepsinin aynı boyuta sahip olması gerekmez.
- Seviye i ve i + 1 arasındaki transferler b_i boyutundaki bloklarla yapılabilirken, i + 1 ve i + 2 arasındaki transferler farklı b_{i+1} blok büyüklükleriyle yapılabilir.

Bellek Hiyerarşisi

Bazı Tanımlamalar

- Bellek hiyerarşisine sahip olmamızın nedeni, çok hızlı bir şekilde davranan ve daha ucuz olan bir belleğe sahip olma isteğimizdir.
- Bunun gerçekleşmesi için, bellek erişimlerinin çoğunun hiyerarşinin üst seviyesinde bulunması gerekir.
  - ✓ Başvurulan adresteki verinin hiyerarşinin üst seviyesinde bulunması durumuna **hit** diyoruz.
- Aksi takdirde, adreslenen nesne hiyerarşinin daha düşük bir seviyesindeyse;
  - ✓ Bir miss'e sahibiz.
  - ✓ miss durumunda adreslenen nesnenin CPU'ya ulaşması daha uzun sürecektir.

Bellek Hiyerarşisi

Bazı Tanımlamalar

- Aranan veri hiyerarşinin üst seviyesinde değilse erişilmek istenen verinin de içerisinde bulunduğu bir blok veri kopyalanarak üst seviyeye getirilir.
- Blok aktarımının nedeni bir sonra erişilecek olan veri büyük olasılıkla şimdi erişilen verinin yakın adresinde olacaktır.
- Blok aktarımı yapmanın olumsuz nedeni ise uzun sürmesi olabilir.
- Belirli bir anda üst seviyede bir alt seviyedeki verilerin bir kısmı bulunabilir.

Bellek Hiyerarşisi

Bazı Tanımlamalar

- Üst seviye doluyken yeni bir veri getirmek gerektiğinde hangi bloğun kaldırılacağını belirlemek için bir yer değiştirme algoritmasına ihtiyaç vardır.
- En yaygın kullanılan yer değiştirme teknikleri:
  - ✓ FIFO (First in First Out): Üst seviyede en uzun süredir yer alan blok çıkartılır
  - ✓ LRU (Least Recently Used): Son zamanlarda en az kullanılan blok çıkartılır
    - Blokların kullanım tarihçesi dikkate alınır
    - Her bloğa atanan yaşlanma sayaçları ilgili bloğa yapılan erişimlerin kaydını tutar.

Bellek Hiyerarşisi

Bazı Tanımlamalar

Hit süresi: Bellek hiyerarşisinin üst seviyesindeki bir nesneye erişmek için geçen süredir, bu süre içerisinde bir hit veya miss olayı gerçekleşir.

- miss durumunda bir miss cezası (miss penalty) vardır çünkü erişilecek olan nesnenin bellekteki düşük seviyeden daha yüksek bir seviyeye getirilmesi gerekir.
- miss cezası iki ayrı zamanı içerir:

Erişim zamanı: Hiyerarşinin alt seviyesindeki bloğun ilk elemanına erişim için geçen süre

Transfer zamanı: Bloğun geri kalanının transferi için geçen süre.

  - Bir miss durumunda bütün bir blok alt seviyeden yenisiyle değiştirilir

Bellek Hiyerarşisi

Bazı Tanımlamalar

Hit rate: İsabet eden bellek erişim oranını ifade eder

Miss rate = 1- hit rate

Belirli bir program ve makine için hit oranı aşağıdaki gibi deneysel olarak belirlenebilir

- ✓ Programı çalıştırın ve belleğe kaç kez erişildiğini sayın (N)
- ✓ Kaç kere hit gerçekleştiğini sayın (N_h)
- ✓ Hit oran $H=N_h / N$

Hiyerarşideki her seviyenin bit maliyetini (C_i) ve kapasitesini (S_i) biliyorsak, bellek hiyerarşisinin maliyeti hesaplanabilir. Sonra bit başına ortalama maliyet aşağıdaki şekilde verilir:

$$C = \frac{C_1 * S_1 + C_2 * S_2 + \cdots . + C_n * S_n}{S_1 + S_2 + \cdots + S_n}$$

Bellek Hiyerarşisi

Bellek hiyerarşisi için performansı tanımlama

- Tasarımcının amacı mümkün olduğunca hızlı bir makineye sahip olmaktır.
- Bellek hiyerarşisine gelince, bellekten mümkün olduğunca küçük bir ortalama erişim süresi istiyoruz.
- Ortalama erişim süresi, hiyerarşinin en üst seviyesindeki belleğin erişim süresinden daha küçük olamaz (t_{A1})
- İki seviyeli bir hafıza hiyerarşisi için ortalama bellek erişim süresi:

$$t_{av} = hit\_time + miss\_rate * miss\_penalty$$

hit süresi temel olarak hiyerarşideki birinci seviyedeki belleğin erişim süresidir t_{A1}

Ayrıca, bir hit mi, yoksa miss mi olduğunu tespit etme zamanıdır

Bellek Hiyerarşisi

Bellek hiyerarşisi için performansı tanımlama

Örnek: İki seviyeli bir bellek hiyerarşisi için aşağıda verilen değerlere göre ortalama erişim süresi nedir?

hit time: 40 ns

Miss penalty: 400 ns

Hit rate: %99

t_{av}=?

Cevap:

Miss_rate=1-hit_rate

Miss_rate=1-0.99=0.01

$t_{av} = hit\_time + miss\_rate * miss\_penalty$

T_{av}= 40+0.01*400=44ns

Hit time'dan %10 daha fazla

Bellek Hiyerarşisi

Bellek hiyerarşisi için performansı tanımlama

Örnek: İki seviyeli bir bellek hiyerarşisi için aşağıda verilen değerlere göre hit rate nedir?

hit time: 1 clock cycle

Miss penalty: 20 clock cycle

t_{av}=1.5

Hit rate: ?

Cevap:

$$\boldsymbol{miss\_rate = \frac{t_{av} - hit\_time}{miss\_penalty}}$$

$$\boldsymbol{miss_{rate} = \frac{1.5 - 1}{20} = 0.0025 = 2.5\%}$$

$$\boldsymbol{hit\_rate = 1 - miss\_rate = \%97.5}$$

Bellek Hiyerarşisi

Bellek hiyerarşisi için performansı tanımlama

Aşağıdaki şekiller verilen 2 seviye bir bellek hiyerarşisi için miss penalty, miss rate ve block boyutu ilişkisini göstermektedir.

- Belli bir blok büyüklüğünün üstünde, miss oranı artmaya başlar; blok boyutu arttıkça hiyerarşinin üst seviyesi çok az blok barındırabilir
- Bellek hiyerarşisi en iyi erişim süresini sağlamalıdır tasarımcı miss rate * miss penalty'nın minimumunu bulmalıdır