1906003052015

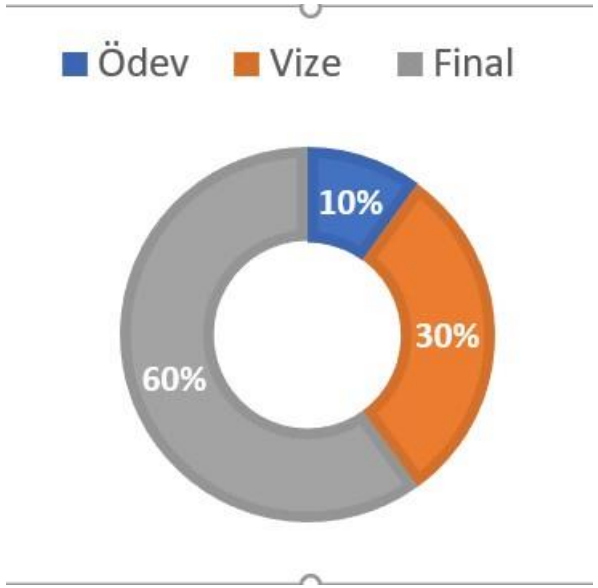
İşletim Sistemleri

Dr. Öğr. Üy. Önder EYECİOĞLU
Bilgisayar Mühendisliği

Giriş

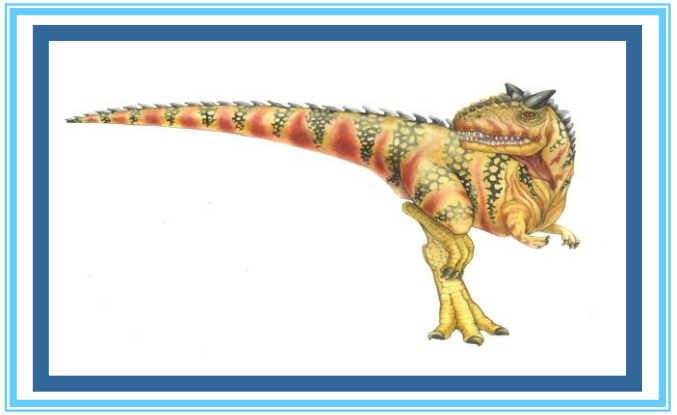
Ders Günü ve Saati:
Çarşamba: 13:00-16:00

- Uygulama Unix (Linux) İşletim sistemi
- Devam zorunluluğu %70
- Uygulamalar C programlama dili üzerinde gerçekleştirilecektir. Öğrencilerden programlama bilgisi beklenmektedir.



HAFTA	KONULAR
Hafta 1	: İşletim sistemlerine giriş, İşletim sistemi stratejileri
Hafta 2	: Sistem çağrıları
Hafta 3	: Görev, görev yönetimi
Hafta 4	: İplikler
Hafta 5	: İş sıralama algoritmaları
Hafta 6	: Görevler arası iletişim ve senkronizasyon
Hafta 7	: Semaforlar, Monitörler ve uygulamaları
Hafta 8	: Vize
Hafta 9	: Kritik Bölge Problemleri
Hafta 10	: Kilitlenme Problemleri
Hafta 11	: Bellek Yönetimi
Hafta 12	: Sayfalama, Segmentasyon
Hafta 13	: Sanal Bellek
Hafta 14	: Dosya sistemi, erişim ve koruma mekanizmaları, Disk planlaması ve Yönetimi
Hafta 15	: Final

Chapter 8: Main Memory



Chapter 8: Memory Management

- Background
- Swapping
- Contiguous Memory Allocation
- Segmentation
- Paging
- Structure of the Page Table
- Example: The Intel 32 and 64-bit Architectures
- Example: ARM Architecture

Objectives

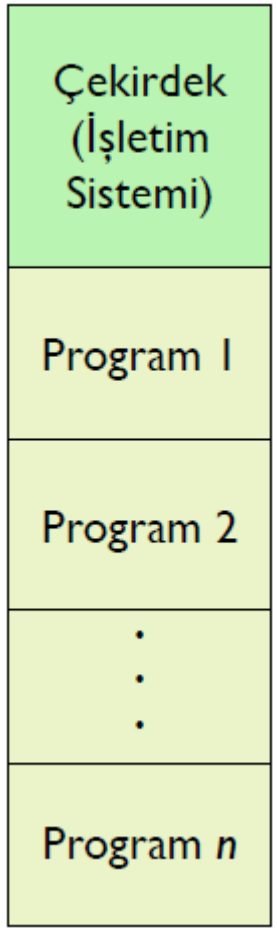
- Bellek, modern bir bilgisayar sisteminin işleyişinin merkezinde yer alır. Bellek, her biri kendi adresine sahip geniş bir bayt dizisinden oluşur. CPU, program sayacının değerine göre komutları bellekten alır. Bu talimatlar, belirli bellek adreslerinden ek yüklemeye ve bu adreslere depolamaya neden olabilir.
- Örneğin, tipik bir talimat yürütme döngüsü, önce bellekten bir talimat getirir. Talimat daha sonra kodu çözülür ve işlenenlerin bellekten alınmasına neden olabilir. Komut işlenenlerde yürütüldükten sonra, sonuçlar bellekte saklanabilir. Bellek birimi yalnızca bellek adresleri akışını görür; bunların nasıl oluşturulduğunu (talimat sayacı, dizin oluşturma, dolaylı adresler, değişmez adresler vb. tarafından) veya ne için olduklarını (talimatlar veya veriler) bilmez. Buna göre, bir programın nasıl bir bellek adresi oluşturduğunu görmezden gelebiliriz. Yalnızca çalışan program tarafından oluşturulan bellek adresleri dizisiyle ilgileniyoruz.

Background

- Programın çalışması için (diskten) belleğe getirilmeli ve bir işlemin içine yerleştirilmelidir
- Ana bellek ve kayıtlar yalnızca CPU'nun doğrudan erişebileceği depolama alanıdır.
- Bellek birimi yalnızca bir adres akışı + okuma istekleri veya adres + veri ve yazma istekleri görürErişimi bir CPU saatine (veya daha azına) kaydeder
- Ana bellek birçok döngü alabilir ve bu da bir durmaya neden olabilir
- Önbellek, ana bellek ve CPU kayıtları arasında otururDoğru çalışmayı sağlamak için gerekli belleğin korunması

Bellek Yönetimi

- An bellek işlemler arasında paylaştırılması gereken önemli diğer bir kaynaktır.
- İşletim sisteminin bir diğer görevi de bellek kullanımını koordine etmek ve yönetmektir.
- Sayfalama (paging) ve bölümleme (segmantation) gibi bellek yönetimi stratejileri bulunur
- İşletim sisteminin bellek hiyararşisini yöneten kısmına «bellek yöneticis» denir.



Bellek Yönetimi

- İşetim sisteminin bellek hiyararşisini yöneten kısmına «bellek yöneticis» denir.
  - Belleğin hangi kısımlarının kullanılıp hangi kısımlarının kullanılmadığı bilgisini tutar.
  - Görevlere ihtiyaç duyduklarında, belleği tahsis eder (allocate) ve ihtiyaç duymadıklarında çeker (deallocate)
  - Bellek yetersis kaldığında, görevleri tutmak için bellek ve disk arasında takas işlemini yönetir.

Bellek Yönetimi

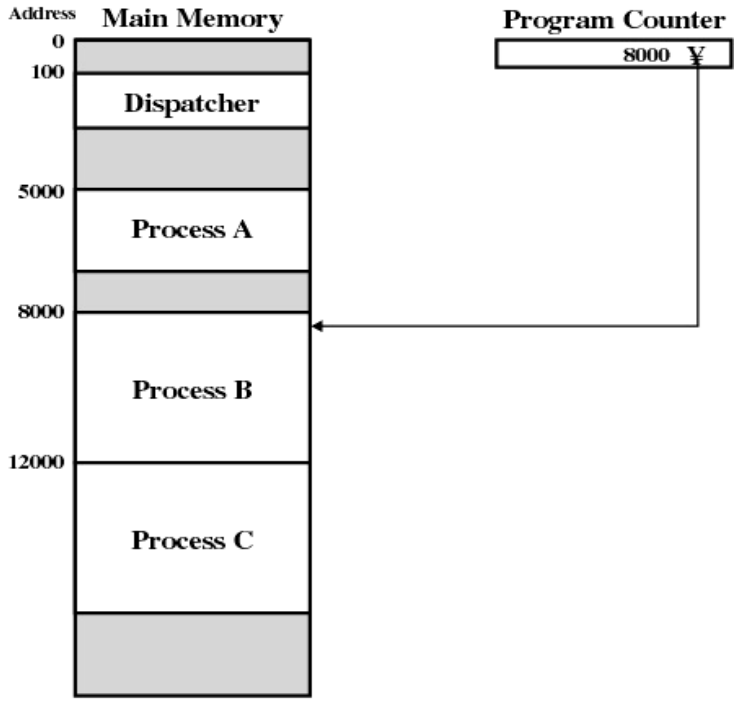
- İşetim sisteminin bellek hiyararşisini yöneten kısmına «bellek yöneticis» denir.
  - Belleğin hangi kısımlarının kullanılıp hangi kısımlarının kullanılmadığı bilgisini tutar.
  - Görevlere ihtiyaç duyduklarında, belleği tahsis eder (allocate) ve ihtiyaç duymadıklarında çeker (deallocate)
  - Bellek yetersis kaldığında, görevleri tutmak için bellek ve disk arasında takas işlemini yönetir.

Bellek Yönetimi

- Bellek her birinin özel adresi olan Word ve baytlardan oluşur.
  - Malloc kütüphane çağrısı
    - Tahsis için kullanılır. (allocation)
    - Yeterli bitişik bellek bulur.
    - Belleği rezerve eder.
  - Free kütüphane çağrısı
    - Serbest belleğin ilk adresini verir
    - Yeniden tahsis (reallocation) için belleği uygun hale getirir.
- Molloc ve free, «brk» sistem çağrısını kullanarak uygulanır.

Bellek Yönetimi

- CPU, programları (instruction) program sayacının değerine göre bellekten getirir.
- Bellek birimi yalnızca bellek akışını görür. Nasıl oluştuklarını ve ya ne için kullanılacaklarını bilmez.
- Bellek birimi, programın nasıl bellek adresi ürettiği ile ilgilenmez.
- Bellek yönetim sistemi iki sınıfa ayrılabilir.,
  - Çalışma (execution) sırasında, görevleri bellek ve disk arasında ileri ve geri taşır. (swapping, paging): Bellek soyutlama- Memory abstraction
  - No memory abstraction.



Donanım

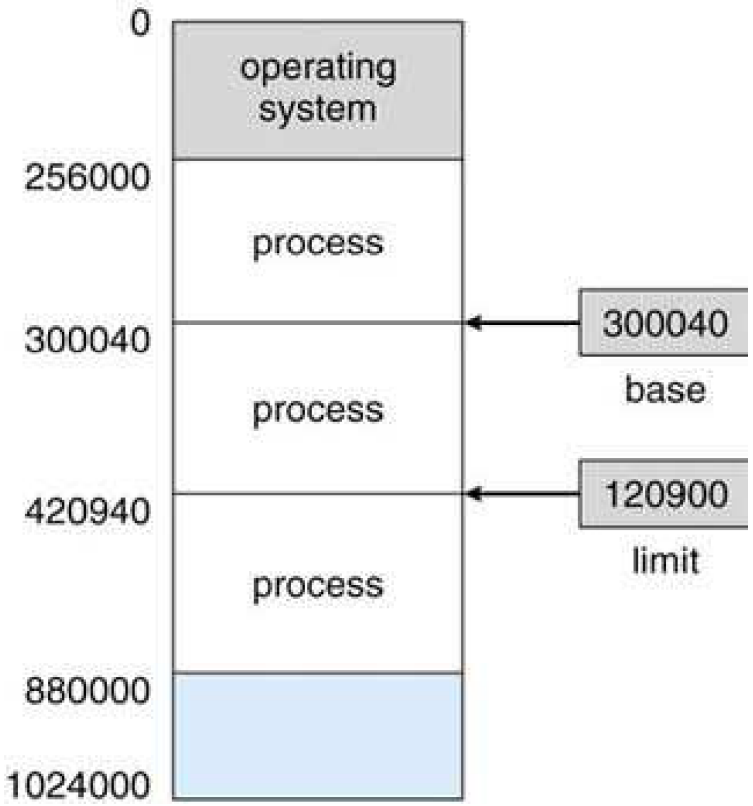
- Ana bellek ve işlemci içersinde bulunan kaydediciler (register) , CPU nun doğrudan erişebilecekleri tek depolama alanlarıdır.
- Buralarda, bellek adreslerini argüman olarak alan makine komutları (instructions) bulunur. (disk adreslerini almazlar.)
- Programın yürütülmesi için gereken komutlar veya datalar bu doğrudan erişimli depolama alanlarınıda bulunmalıdır.

Donanım

- Kaydedicilere bir CPU saatinin bir çevriminde erişilebilir.
- Ana bellek yönetiminde, bir işleme bellek veri yolu ile ulaşılabilir. Bellek erişimi işlemci saatinin birçok döngüleri sürebilir.
- Bunun için ana bellek ile CPU arasına hızlı geçici bellek (cache) eklemek gerekir.
- Bellek erişim hızının yanında OS ni kullanıcı görevlerinden ve kullanıcı görevlerini diğer kullanıcı görevlerinden korumak gereklidir.

Base and Limit Registers

- Bunu donanımın sağlaması gereklidir. Bunun için öncelikle, her görevin, kendine ait ayrı bellek alanına sahip olduğundan emin olmak gerekir.

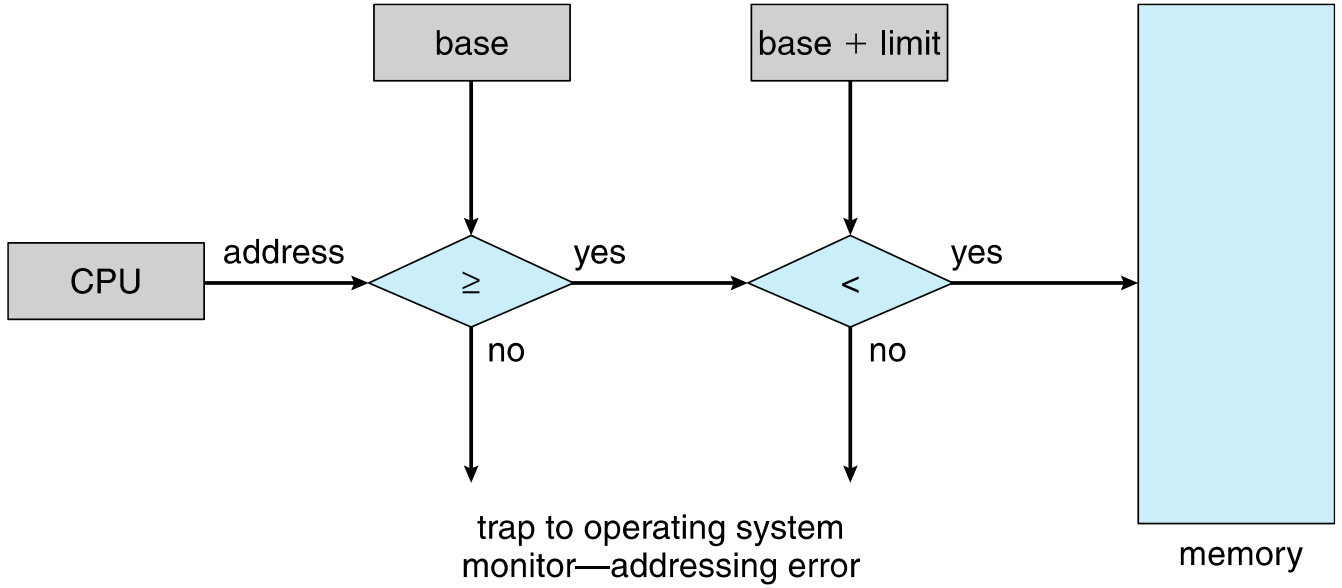


Bu koruma, «base» ve «limit» kaydedicileri ile gerçekleştirilir.
«base» kaydedici uygun endüşük bellek adresini belirler
«limit» kaydedici aralığın boyutunu belirler.

Bu koruma CPU donanımının, kullanıcı modda üretilen her bellek adresinin, kaydediciler ile karşılatırılması ile yapılır.

Hardware Address Protection

- Kullanıcı modundaki her hangi bir programın, OS veya diğer kullanıcı programlarının oluşturduğu bellek tuzak (trap) alanına erişmesi için yapılan girişim ölümcül hata girişimi (fatal error) olarak davranır.



Address Binding

- Genellikle, bir program bir diskte ikili yürütülebilir dosya olarak bulunur.
- Çalıştırmak için, programın belleğe alınması ve mevcut bir CPU üzerinde yürütülmeye uygun hale geldiği bir süreç bağlamına yerleştirilmesi gerekir.
- İşlem yürütülürken, bellekten talimatlara ve verilere erişir.
- Sonunda süreç sona erer ve belleği diğer süreçler tarafından kullanılmak üzere geri alınır.
- Çoğu sistem, bir kullanıcı işleminin fiziksel belleğin herhangi bir bölümünde kalmasına izin verir.
- Bu nedenle, bilgisayarın adres alanı 00000'den başlasa da, kullanıcı işleminin ilk adresinin 00000 olması gerekmez.

Address Binding

- Çoğu durumda, bir kullanıcı programı yürütülmeden önce, bazıları isteğe bağlı olabilen birkaç adımdan geçer (Şekil 9.3).
- Bu adımlar sırasında adresler farklı şekillerde gösterilebilir.
- Kaynak programdaki adresler genellikle semboliktir (değişken sayısı gibi).
- Derleyici tipik olarak bu sembolik adresleri yeniden yerleştirilebilir adreslere bağlar ("bu modülün başlangıcından itibaren 14 bayt" gibi).
- Bağlayıcı veya yükleyici yeri değiştirilebilen adresleri mutlak adreslere (74014 gibi) bağlar.
- Her bağlama, bir adres alanından diğerine bir eşlemedir.

Multistep Processing of a User Program

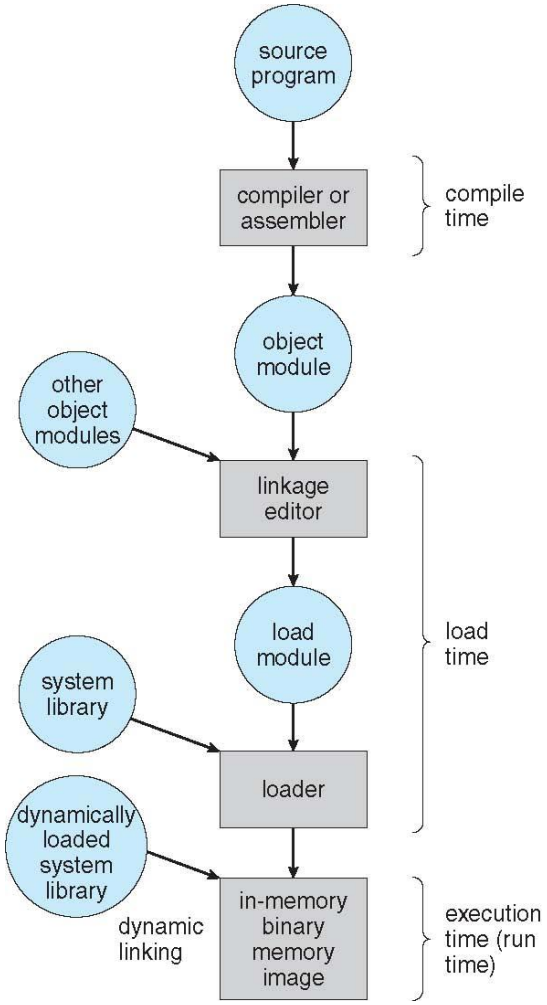


Figure 9.3 Multistep processing of a user program

Binding of Instructions and Data to Memory

Talimatların ve verilerin bellek adreslerine adres bağlaması üç farklı aşamada gerçekleşebilir

- **Compile time**: Derleme zamanında işlemin bellekte nerede kalacağını biliyorsanız, mutlak kod üretilebilir. Örneğin, bir kullanıcı işleminin R konumundan başlayacağını biliyorsanız, oluşturulan derleyici kodu o konumdan başlayacak ve oradan yukarıya doğru genişleyecektir. Daha sonra başlangıç konumu değişirse, bu kodu yeniden derlemek gerekecektir.
- **Load time**: Derleme zamanında işlemin bellekte nerede kalacağı bilinmiyorsa, derleyicinin yeri değiştirilebilen kod oluşturması gerekir. Bu durumda, son bağlama yükleme süresine kadar ertelenir. Başlangıç adresi değişirse, bu değiştirilen değeri dahil etmek için yalnızca kullanıcı kodunu yeniden yüklememiz gerekir.
- **Execution time**: İşlem yürütülürken bir bellek bölümünden diğerine taşınabilirse, bağlama çalışma zamanına kadar ertelenmelidir. Bu şemanın çalışması için özel donanım mevcut olmalıdır. Çoğu işletim sistemi bu yöntemi kullanır.
  - Need hardware support for address maps (e.g., base and limit registers)

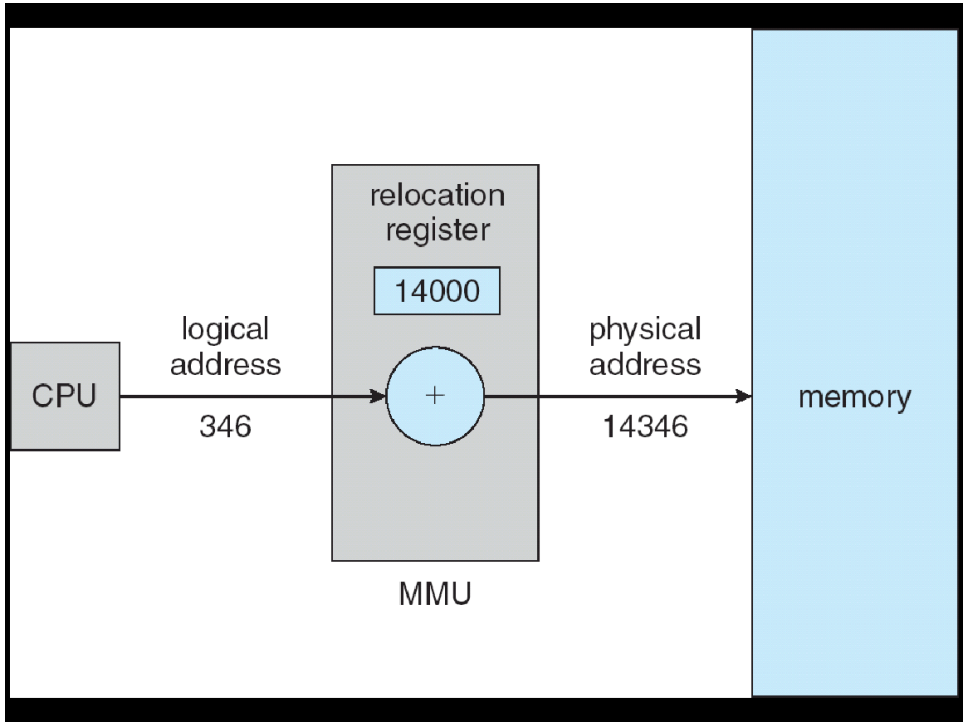
Logical vs. Physical Address Space

- The concept of a logical address space that is bound to a separate **physical address space** is central to proper memory management
  - **Logical address** – generated by the CPU; also referred to as **virtual address**
  - **Physical address** – address seen by the memory unit
- Logical and physical addresses are the same in compile-time and load-time address-binding schemes; logical (virtual) and physical addresses differ in execution-time address-binding scheme
- **Logical address space** is the set of all logical addresses generated by a program
- **Physical address space** is the set of all physical addresses generated by a program

Mantıksal ve Fiziksel Adres Uzayı

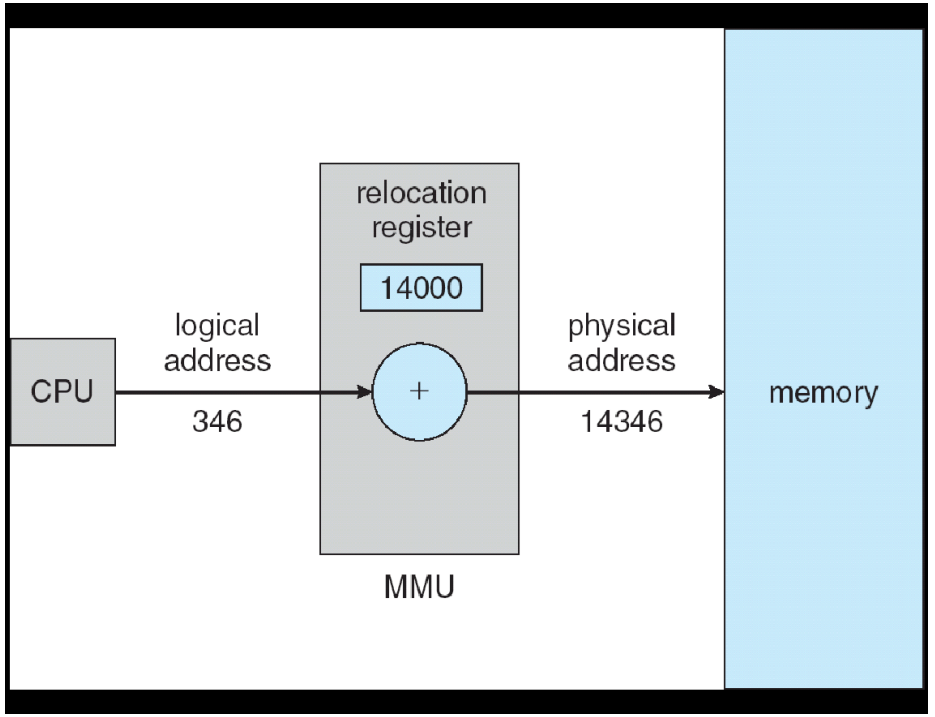
- **Mantıksal adres (Logical adress):** CPU tarafından üretilir.(Sanal adres)
- **Fiziksel adres (Physical adress):** Bellek ünitesi tarafından görünen adres.

- Fiziksel ve mantıksal adresler, derleme ve yükleme zamanında aynıdır. Yürütme zmanında farklılaşırlar.



Mantıksal ve Fiziksel Adres Uzayı

- ►Mantıksal
- – belleğe erişimde kullanılan adres gerçek fiziksel
- adreslerden bağımsız
- – adres dönüşümü gerekir
- ►Bağıl
- – adres bilinen bir noktaya göre bağıl verilir
- – genellikle bu nokta prosesin başıdır
- ►Fiziksel
- – ana bellekteki gerçek konum adresi
- Fiziksel ve mantıksal adresler, derleme ve yükleme zamanında aynıdır. Yürütme zmanında farklılaşırlar.



Memory-Management Unit (MMU)

- Sanal adreslerden fiziksel adreslere çalışma zamanı eşlemesi, bellek yönetim birimi (MMU) adı verilen bir donanım aygıtı tarafından yapılır (Şekil 9.4).

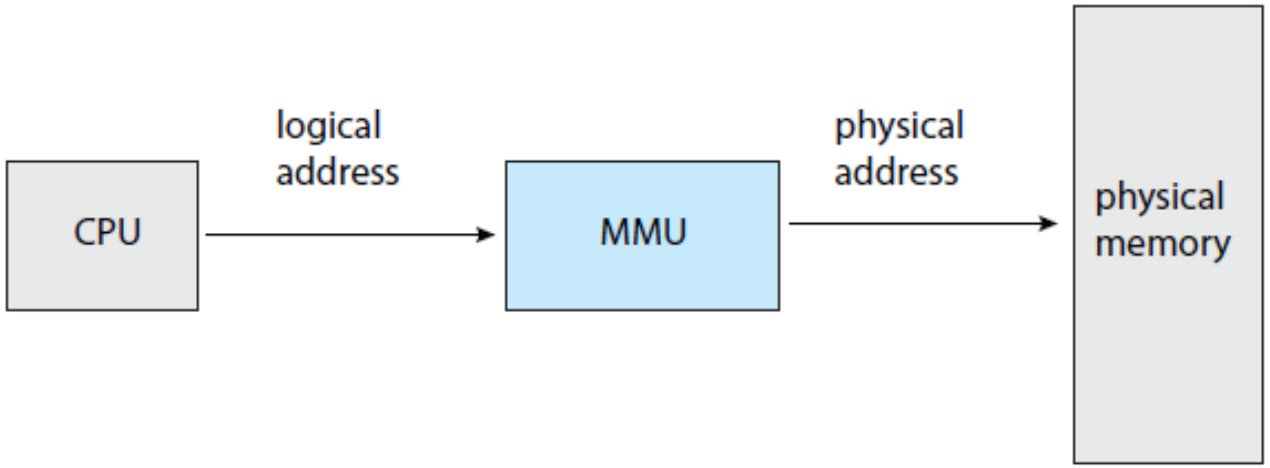
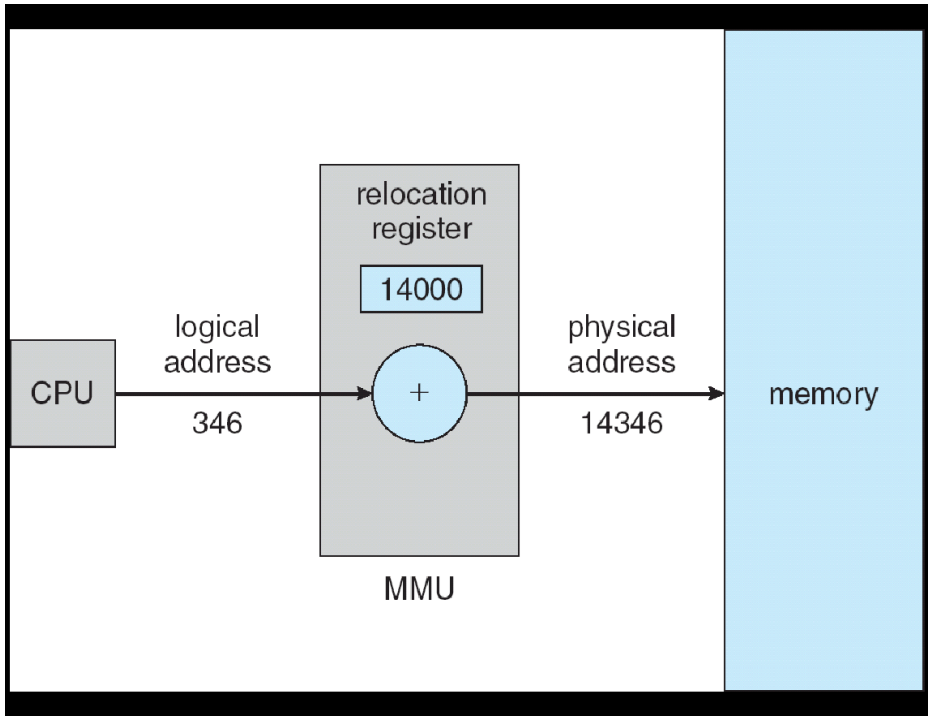


Figure 9.4 Memory management unit (MMU).

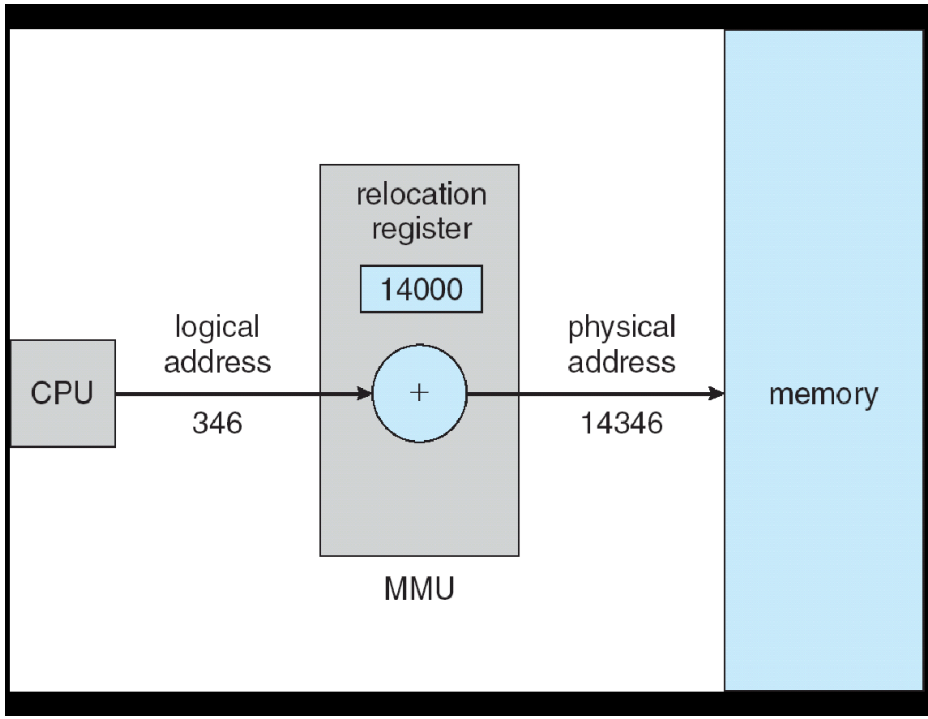
Memory-Management Unit (MMU)

- Bu eşlemeyi temel kayıt şemasının bir genellemesi olan basit bir MMU şemasıyla gösteriyoruz. Temel kayıt, artık bir yer değiştirme kaydı olarak adlandırılmaktadır. Yer değiştirme kaydındaki değer, adres belleğe gönderildiğinde bir kullanıcı işlemi tarafından oluşturulan her adrese eklenir (bkz. Şekil 9.5). Örneğin, baz 14000'deyse, kullanıcı tarafından konum 0'a yönelik bir girişim, dinamik olarak konum 14000'e yeniden konumlandırılır; 346 konumuna bir erişim, 14346 konumuna eşlenir.



Memory-Management Unit (MMU)

- Kullanıcı programı hiçbir zaman gerçek fiziksel adreslere erişmez. Program, konum 346'ya bir işaretçi oluşturabilir, onu bellekte saklayabilir, değiştirebilir ve diğer adreslerle karşılaştırabilir - hepsi 346 sayısı olarak. Yalnızca bir bellek adresi olarak kullanıldığında (belki dolaylı bir yüklemede veya depoda) temel kayıt defterine göre mi yer değiştiriyor? Kullanıcı programı mantıksal adreslerle ilgilenir. Bellek eşleme donanımı, mantıksal adresleri fiziksel adreslere dönüştürür. Başvurulan bir bellek adresinin son konumu, başvuru yapılana kadar belirlenmez.



Memory-Management Unit (MMU)

- Sanal adreslerden fiziksel adreslere çalışma zamanı eşlemesi, bellek yönetim birimi (MMU) adı verilen bir donanım aygıtı tarafından yapılır (Şekil 9.4).
- Başlamak için, yer değiştirme kaydındaki değerin, belleğe gönderildiği anda bir kullanıcı işlemi tarafından oluşturulan her adrese eklendiği basit bir şema düşünün
  - Temel kayıt artık **relocation register**olarak adlandırılıyor
  - MS-DOS on Intel 80x86 used 4 relocation registers
- Kullanıcı programı mantıksal adreslerle ilgilenir. gerçek fiziksel adresleri asla görmez
  - Yürütme zamanı bağlaması, bellekteki konuma başvuru yapıldığında gerçekleşir
  - Mantıksal adresler Fiziksel adreslere bağlıdır.

Bellek Yönetim Birimi (MMU)

- **Fiziksel adrese sanal bir plan (map) oluşturan donanımdır.**
- **Kullanıcı programları mantıksal adres ile ilgilidir. Fiziksel adresleri görmezler.**

```
$ pmap -x 3901 | less   3901:   /opt/firefox/firefox
                        Address           Kbytes     RSS   Dirty Mode   Mapping
                        0000000000400000       0      76       0 r-x--  firefox
                        0000000000615000       0       4       4 rw---  firefox
                        00007f77c918c000       0       0       0 -----    [ anon ]
                        00007f77c918d000       0      12      12 rw---    [ anon ]
                        00007f77c998d000       0     168       0 r----  icon-theme.cache
                        00007f77cccfc000       0       0       0 -----    [ anon ]
                        00007f77cccfd000       0      12      12 rw---    [ anon ]
                        00007f77ce200000       0    1024    1024 rw---    [ anon ]
                        00007f77ce3b6000       0     168       0 r----  icon-theme.cache
                        00007f77cefff000       0       0       0 -----    [ anon ]
                        00007f77cf000000       0    1044    1044 rw---    [ anon ]
```

Dynamic relocation using a relocation register

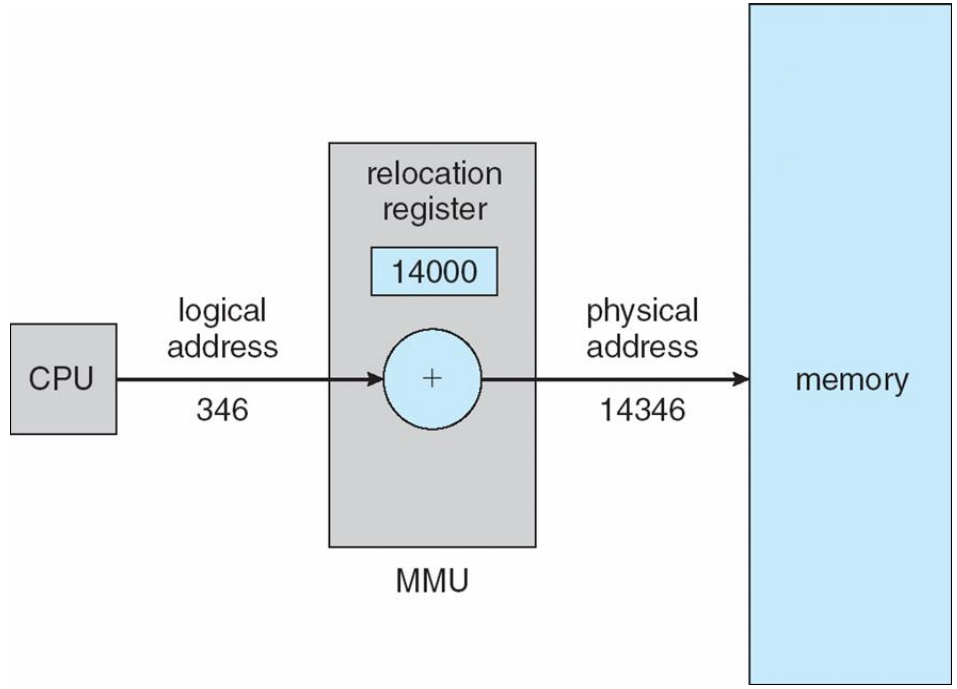
- Daha iyi bellek alanı kullanımı elde etmek için dinamik yüklemeyi kullanabiliriz.
- Dinamik yükleme ile bir rutin çağrılana kadar yüklenmez. Tüm rutinler, yeniden yerleştirilebilir bir yükleme biçiminde diskte tutulur.
- Ana program belleğe yüklenir ve yürütülür.
- Bir rutinin başka bir rutini çağırması gerektiğinde, çağıran rutin önce diğer rutinin yüklenip yüklenmediğini kontrol eder.
- Değilse, istenen rutini belleğe yüklemek ve programın adres tablolarını bu değişikliği yansıtacak şekilde güncellemek için yeniden yerleştirilebilir bağlantı yükleyici çağrılır. Ardından kontrol, yeni yüklenen rutine geçirilir.

Dynamic relocation using a relocation register

- Dinamik yüklemenin avantajı, bir rutinin yalnızca ihtiyaç duyulduğunda yüklenmesidir.
- Bu yöntem, hata rutinleri gibi seyrek olarak meydana gelen durumları işlemek için büyük miktarda kod gerektiğinde özellikle yararlıdır.
- Böyle bir durumda, toplam program boyutu büyük olsa da, kullanılan (ve dolayısıyla yüklenen) kısım çok daha küçük olabilir.
- Dinamik yükleme, işletim sisteminden özel destek gerektirmez. Programlarını böyle bir yöntemden yararlanmak için tasarlamak kullanıcıların sorumluluğundadır.
- Ancak işletim sistemleri, dinamik yüklemeyi uygulamak için kitaplık rutinleri sağlayarak programcıya yardımcı olabilir.

Dynamic relocation using a relocation register

- Routine is not loaded until it is called
- Better memory-space utilization; unused routine is never loaded
- All routines kept on disk in relocatable load format
- Useful when large amounts of code are needed to handle infrequently occurring cases
- No special support from the operating system is required
  - Implemented through program design
  - OS can help by providing libraries to implement dynamic loading



Dynamic Linking

- **Static linking** – yükleyici tarafından ikili program görüntüsünde birleştirilen sistem kitaplıkları ve program kodu
- **Dynamic linking** –bağlama yürütme zamanına kadar ertelenir
- Stub replaces itself with the address of the routine, and executes the routine
- İşletim sistemi, rutinin işlemlerin bellek adresinde olup olmadığını kontrol eder.
  - Adres alanında değilse, adres alanına ekler
  - Dinamik bağlantı özellikle kitaplıklar için kullanışlıdır
- System, **shared libraries** olarak da bilinir.

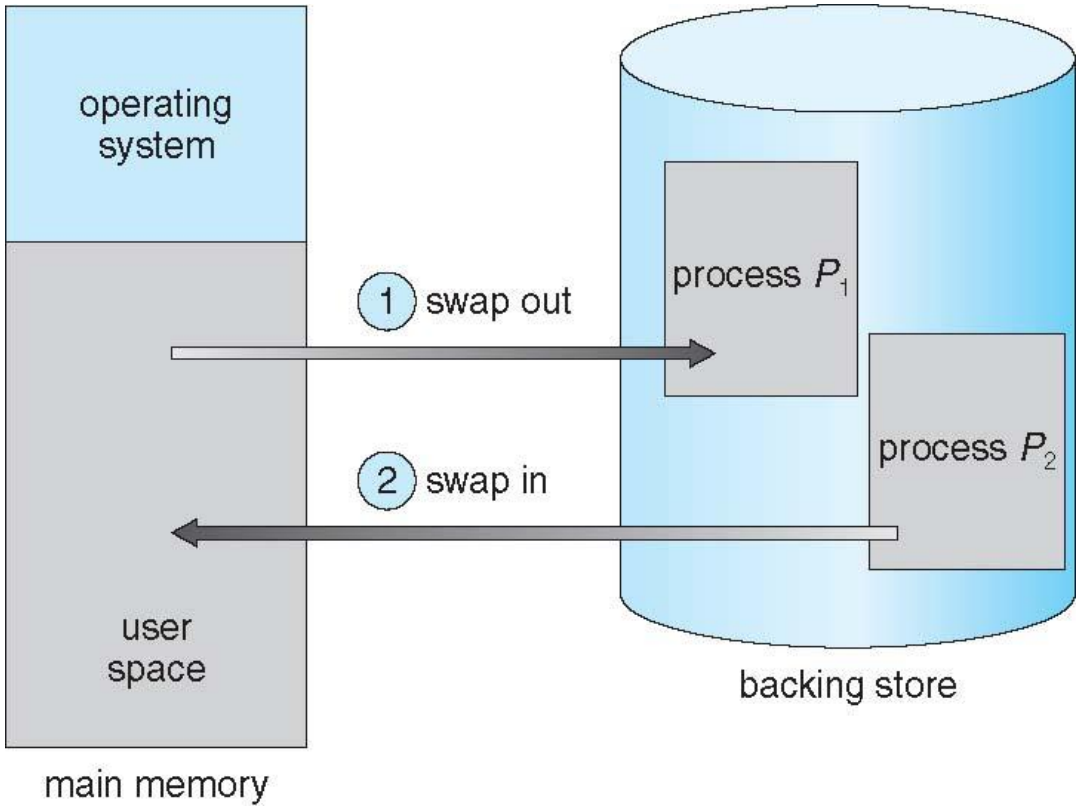
Swapping

- A process can be **swapped** temporarily out of memory to a backing store, and then brought back into memory for continued execution
  - Total physical memory space of processes can exceed physical memory
- **Backing store** – fast disk large enough to accommodate copies of all memory images for all users; must provide direct access to these memory images
- **Roll out, roll in** – swapping variant used for priority-based scheduling algorithms; lower-priority process is swapped out so higher-priority process can be loaded and executed
- Major part of swap time is transfer time; total transfer time is directly proportional to the amount of memory swapped
- System maintains a **ready queue** of ready-to-run processes which have memory images on disk

Swapping (Cont.)

- Does the swapped out process need to swap back in to same physical addresses?
- Depends on address binding method
  - Plus consider pending I/O to / from process memory space
- Modified versions of swapping are found on many systems (i.e., UNIX, Linux, and Windows)
  - Swapping normally disabled
  - Started if more than threshold amount of memory allocated
  - Disabled again once memory demand reduced below threshold

Schematic View of Swapping



Context Switch Time including Swapping

- If next processes to be put on CPU is not in memory, need to swap out a process and swap in target process
- Context switch time can then be very high
- 100MB process swapping to hard disk with transfer rate of 50MB/sec
  - Swap out time of 2000 ms
  - Plus swap in of same sized process
  - Total context switch swapping component time of 4000ms (4 seconds)
- Can reduce if reduce size of memory swapped – by knowing how much memory really being used
  - System calls to inform OS of memory use via `request_memory()` and `release_memory()`

Context Switch Time and Swapping (Cont.)

- Other constraints as well on swapping
  - Pending I/O – can't swap out as I/O would occur to wrong process
  - Or always transfer I/O to kernel space, then to I/O device
    - Known as **double buffering**, adds overhead
- Standard swapping not used in modern operating systems
  - But modified version common
    - Swap only when free memory extremely low