الملزمة السابعة

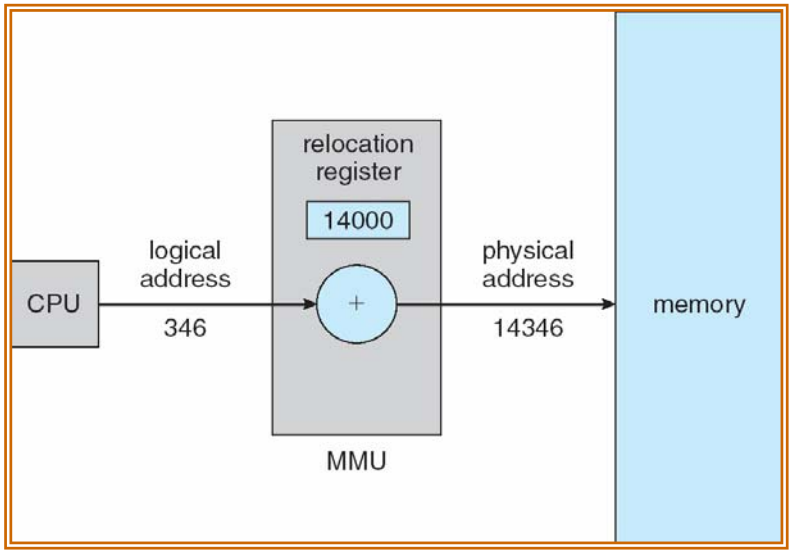
Memory Management

Logical vs. Physical Address

- ***Logical(virtual) address*** : generated by the CPU.
- ***Physical address*** : address seen by the memory unit.
- Logical and physical addresses are the same in ***compile-time*** and ***load-time*** but they differ in ***execution-time***.

Memory-Management Unit (MMU)

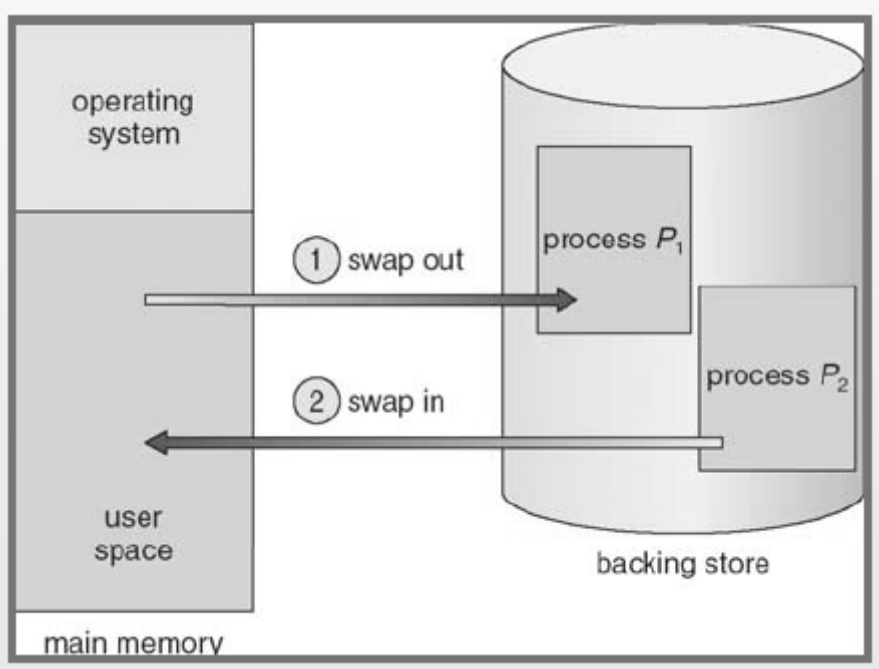
- Hardware device that maps virtual to physical address.
- In MMU scheme, the value in the relocation register is added to every address generated by a user process at the time it is sent to memory.
- The user program deals with logical addresses; it never sees real physical addresses.



Dynamic relocation using a relocation register

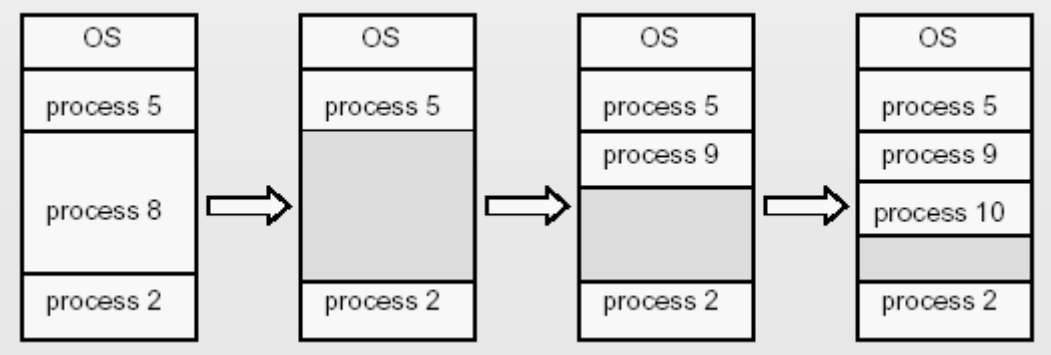
Swapping

A process can be swapped temporarily out of memory to a backing store, and then brought back into memory for continued execution. lower-priority process is swapped out so higher-priority process can be loaded and executed.



Contiguous Memory Allocation

O.S keeps a table indicating which parts of memory are free (holes) and which are allocated.



- In general, there is at any time a set of holes, of various sizes, scattered through out memory. When a process arrives and needs memory, O.S search this set for a hole large enough for this process.

Dynamic Storage-Allocation Problem

How to satisfy a request of size n from a list of free holes?

There are many solutions:

1-***First-fit:*** Allocate the first hole that is big enough.

2-***Best-fit:*** Allocate the smallest hole that is big enough.

3- ***Worst-fit:*** Allocate the hole.

- First-fit and best-fit better than worst-fit in terms of speed and storage utilization.

Fragmentation

The above algorithms suffer from ***External Fragmentation*** which exists when enough total memory space exists to satisfy a request, but is not contiguous, i.e. ***storage is fragmented into a large number of small holes.***

- One solution to the problem of external fragmentation is called ***Compaction***. Shuffle memory contents to place all free memory together in one large block. This scheme is expansive.
- Another solution to the external fragmentation is to ***permit the physical address space of a process to be noncontiguous***. There are two techniques for this solution:
  1- Paging
  2- Segmentation

Paging

- Is a scheme that permits the physical address space of a process can be noncontiguous.

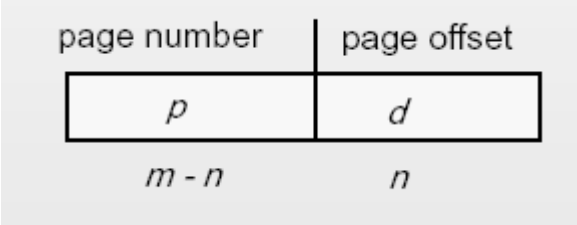
- Physical memory(RAM) is broken into fixed-sized block called ***Frames*** (size is power of 2, between 512 bytes and 8,192 bytes).
- Divide logical memory(Hard) into blocks of the same size called ***Pages***
- To run a program of size n pages, need to find n free frames and load program.

Address Translation

Every address generated by CPU is divided into:

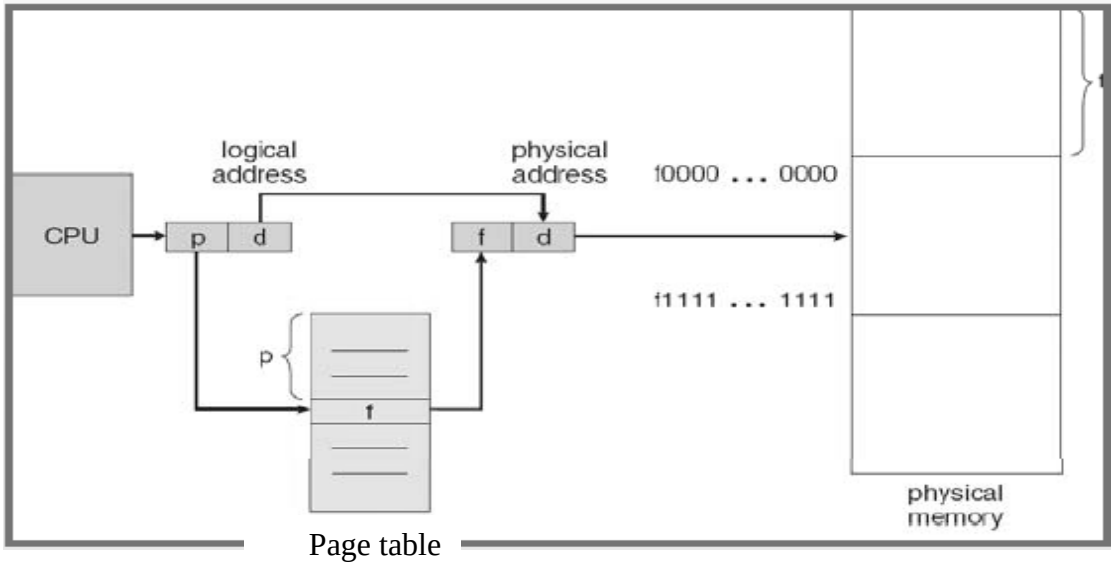
* ***page number*** (P): used as index into Page Table. Which consists the base address of each page in physical memory.

* ***page offset*** (d): which is combined with the base address to define the physical memory address that is sent to the memory unit.

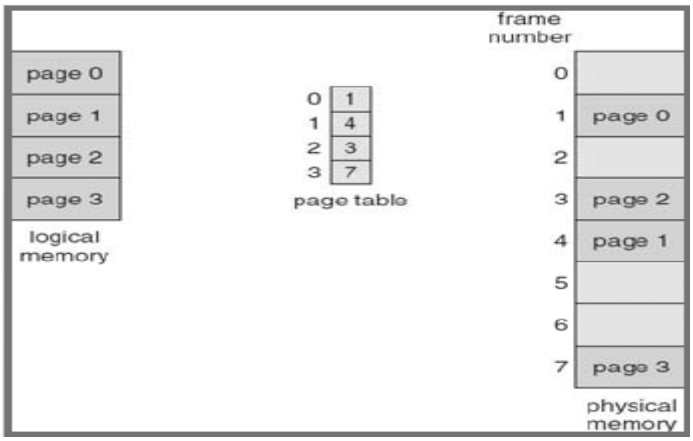


For 2^m logical address space, and 2^n page size

Paging Hardware



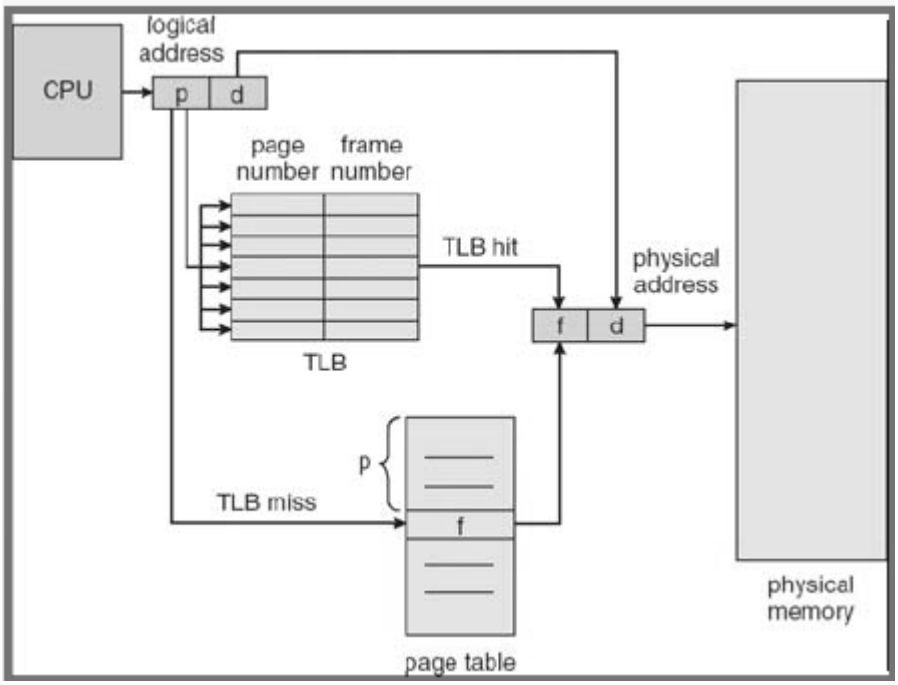
Paging Model



- Paging scheme have no external fragmentation, since any free frame can be allocated to a process that needs it.
- But it have some ***Internal Fragmentation***, if a process needs n pages plus one byte, then it would be allocate n+1 frames.
- The user program in paging is scattered in physical memory.
- Page table is provided by O.S for ***each*** process.

Implementation of Page Table

- Page table may be implemented as a set of registers (small page table).
- If page table is very large, it is kept in m.m. but this required two memory accesses. One for the page table and one for the data.
- The two memory access problem can be solved by the use of a special fast-cache called ***associative memory*** or ***translation look-aside buffers*** (***TLBs***).

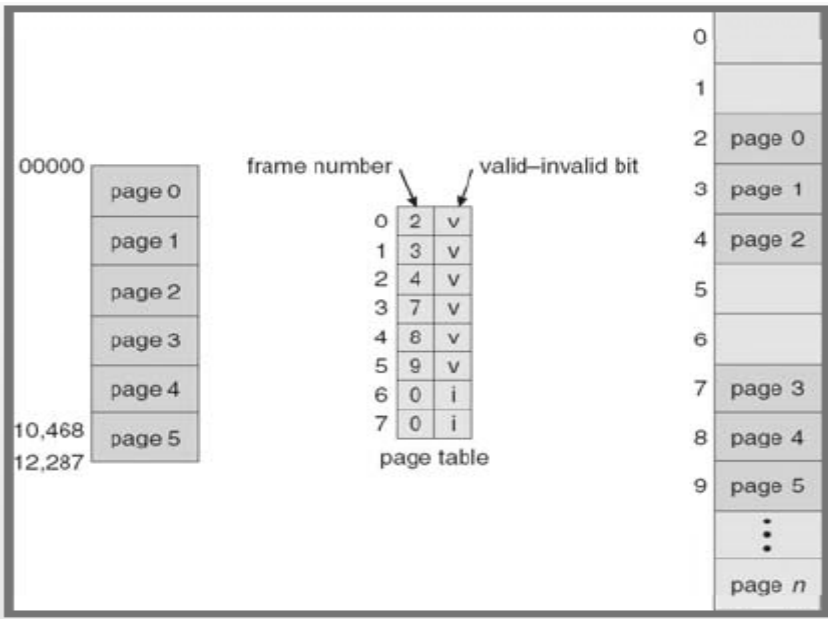


Paging Hardware With TLB

Memory Protection

Memory protection implemented by associating protection bit with each frame

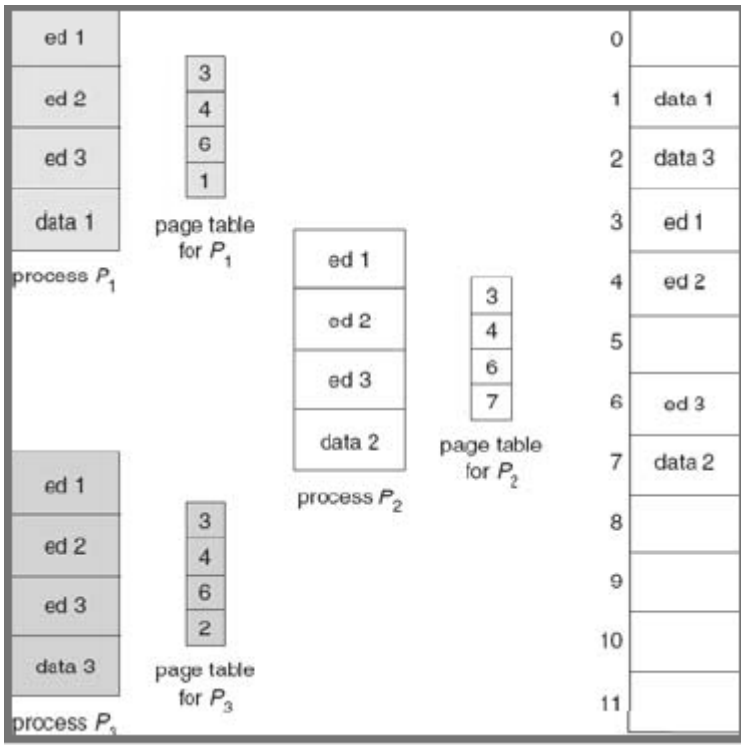
- Valid-invalid bit attached to each entry in the page table: "valid" indicates that the associated page is in the process' logical address space, and is thus a legal page "invalid" indicates that the page is not in the process logical address space.



Valid (v) or Invalid (i) bit in a page table

Shared Pages

- One advantage of paging is the possibility of sharing.
- ***Example***: consider a system that support 40 users, each whom execute a text editor. If the text editor consist of 150k of code and 50k of data. So, we need 8000k for 40 user without sharing. If the code is sharable, we need a copy of the editor (150k), plus 40 copies of the 50k of data. So, the total space is now 2150k, instead of 8000k.



Shared pages example

Two-Level Page Table Scheme

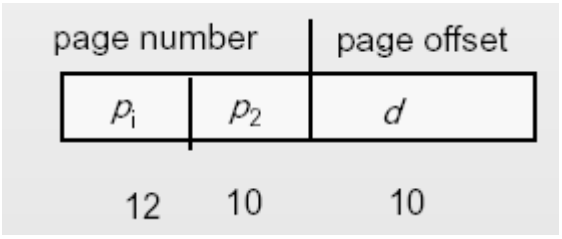
A logical address (on32-bit machine with 1K page size) is divided into:

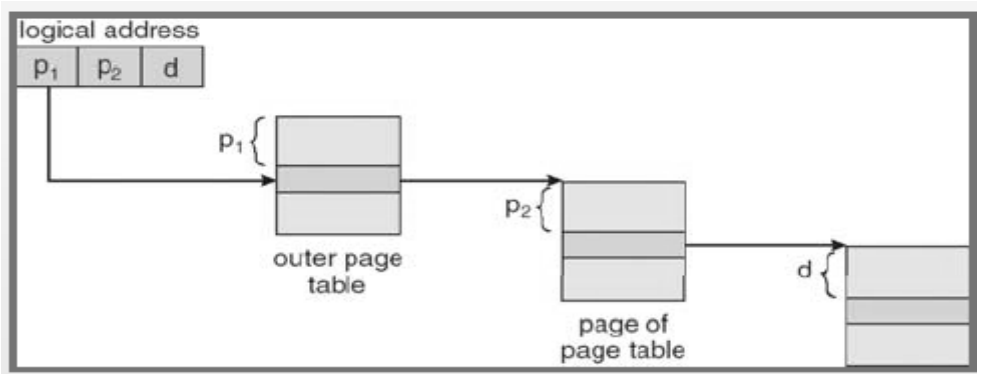
- A page number consisting of 22 bits.
- A page offset consisting of 10 bits.

Since the page table is paged, the page number is further divided into :

- a 12-bit page number.

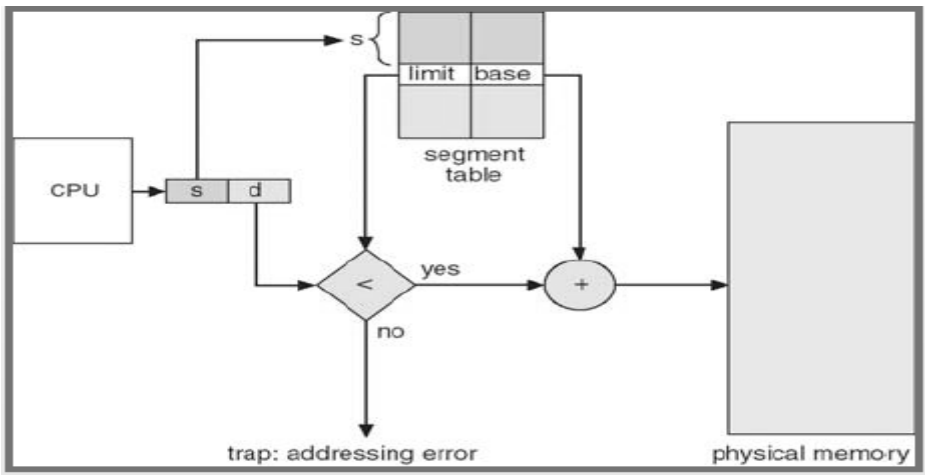
- a 10-bit page offset. Thus, a logical address is as follows:



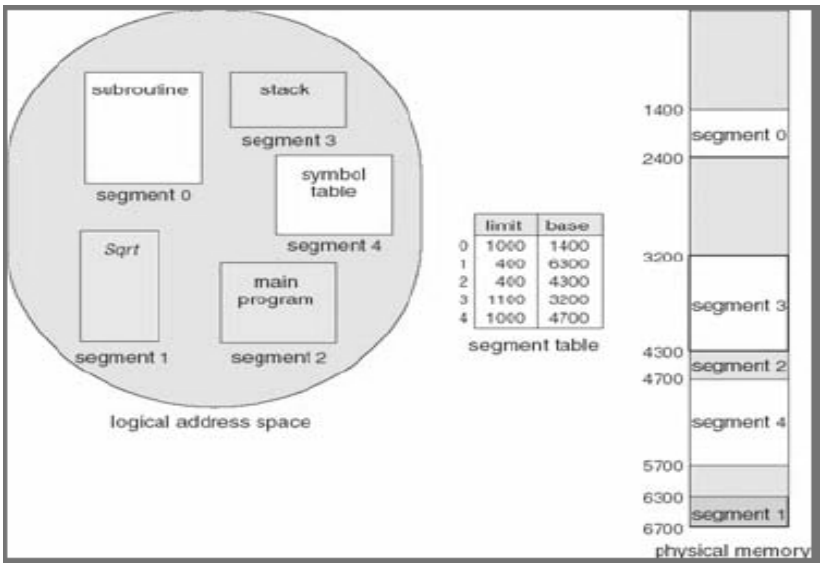


Segmentation

- User program is a collection of segments (subroutines, functions, tables, arrays, stacks, objects,…).
- Each esgment has a name and length. So, the logical address consist here of two fields: < Sgment-number (S),offset (d) >
- Here we use ***Segment Table***. Each entry in this table has:
  - ***base:*** contains the starting physical address where the segments reside in memory.
  - ***limit:*** specifies the length of the segment.



Segmentation Hardware



Segmentation Example

- segments have variable lengths.
- Segmentation suffer from external framgementation.
- Protection and sharing in segmenation is similar to paging.
- To avoid the problems of paging and segmentation, many O.S compine them together.

H.W$_1$): Compare between paging and segmentation.

H.W$_2$): Explain the differnce between external and internal fragementation.

H.W$_3$): Given memory partitions of 100k, 500k, 200k, 300k, and 600k (in order), how would each of the first-fit, best-fit, and worst-fit algorithms place processes of 212k, 417k, 112k, and 426k (in order)? Which algorithm makes the most efficient use of memory?