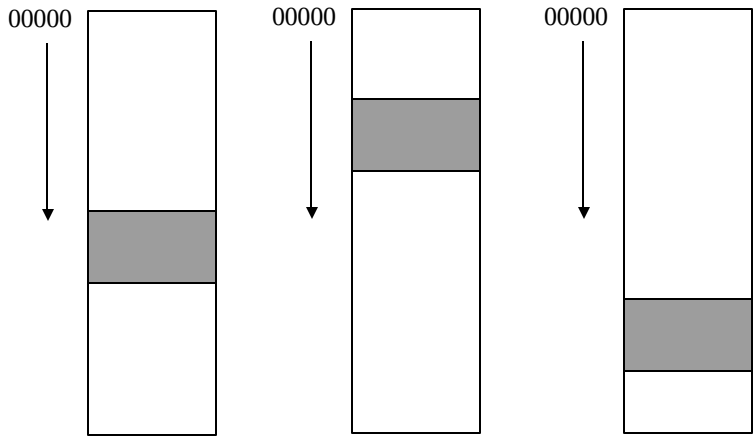
CSC 453 Operating Systems

Lecture 8 : Memory Management

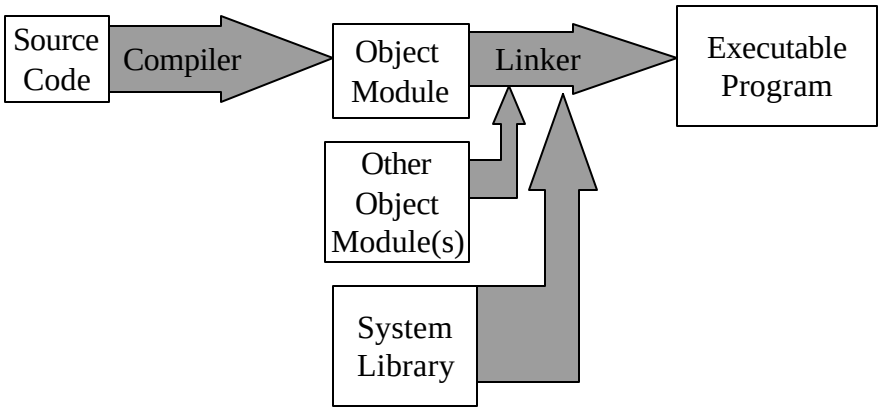
Memory Is A Resource

- Main memory is an array of words or bytes that can be accessed directly by their own unique addresses.
- Executing an instruction will involve accessing memory several times to :
  - fetch the instruction
  - fetch operand(s)
  - store the result
- Memory allocated to one process cannot be used by another process (unless the process allows it).

The Same Program May Be Loaded Into Different Memory Locations



Preparing A Source Program For Execution



Address Binding

Address binding can occur at:

- Compile time
  - when the program is translated, absolute addresses are generated.
  - This assumes that the compiler knows where in memory the program will be loaded.
- Load time
  - The compiler generates relocatable addresses.
  - The binding occurs when the program is loaded
- Execution time
  - If the process can moved during execution, address binding is delayed until run time.

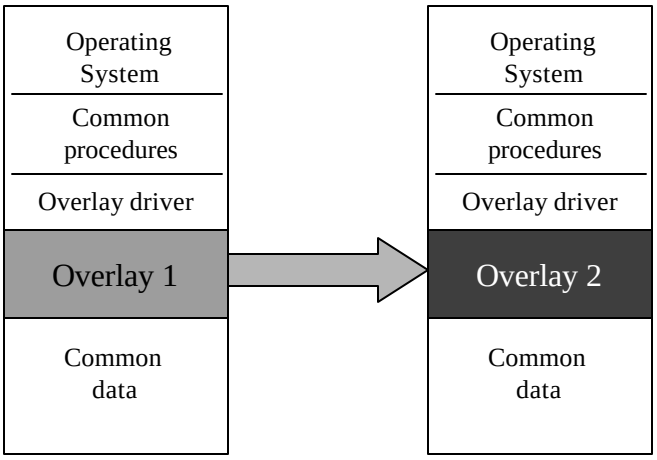
Dynamic Loading

- The main program is loaded into memory and it is executed. All other routines are stored on disk with relocatable addresses.
- When a routine is called, it is read into memory and the program's address table is updated to show the change.
- This allows large programs to run using only a fraction of the memory that they would need if the entire program is stored in memory.
- Users design their programs to use this approach, so no special operating system support is needed.

Dynamic Linking

- Most programs are statically linked, i.e. the libraries are combined with other modules to form the executable module.
- Some systems support dynamic linking where a routine called a stub is linked for every library routine instead of the actual code.
- When the procedure is called, the stub is executed.
  - If the procedure is not in memory, it loads it into memory.
  - The procedure replaces itself with the routine's address.
- Dynamic linking requires operating system support because the procedures may be in an area of memory belonging to another process.

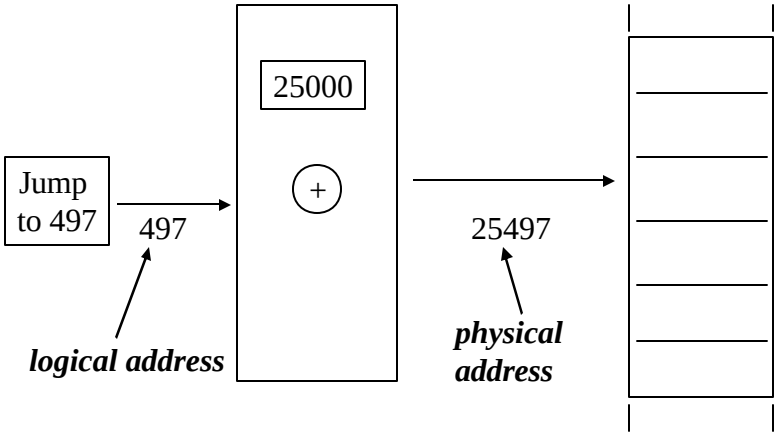
Overlays



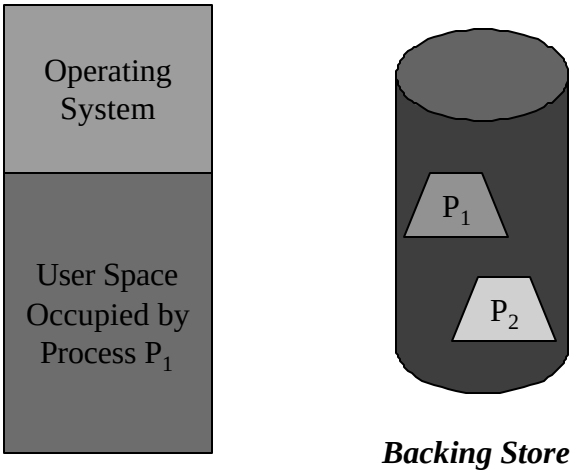
Logical Vs. Physical Address

- ***Logical Address*** – The address used by the instructions to load and store data, jump to other instructions, etc.
- ***Physical Address*** – Used by the hardware to load and store data, fetch instructions, etc.

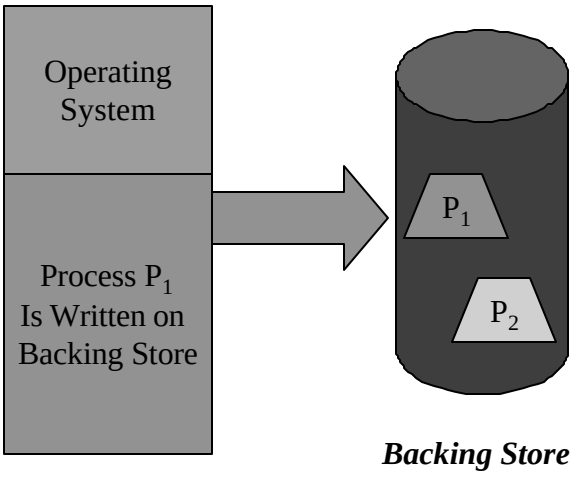
Relocatable Addressing



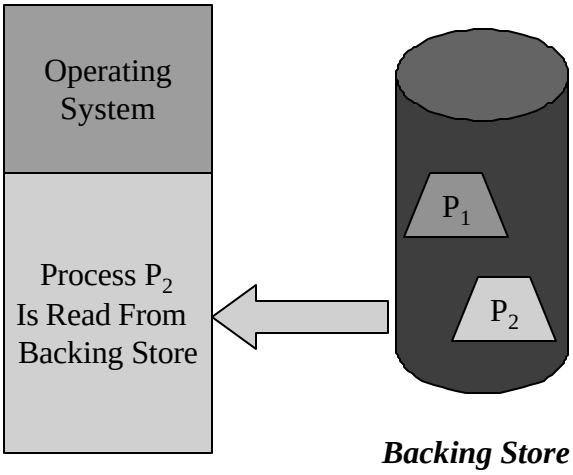
What Is Swapping?



Swapping Out A Process



Swapping In A Process



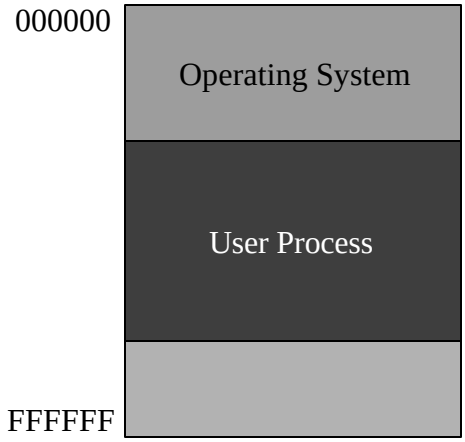
Bare Machine

No Restrictions On Memory Use
-
No Operating System Services

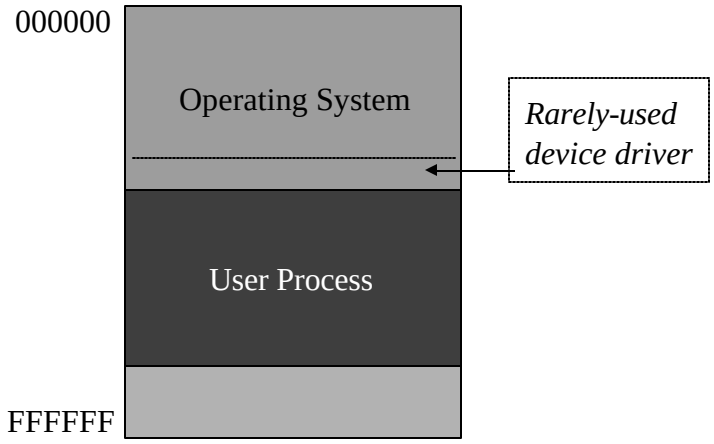
Organizing Memory

- Main memory is comprised of two partitions:
  - Operating system
  - User processes
- The operating system can be at the high end or low end of memory – the location of the interrupt vector is largely responsible for where the operating system will reside.

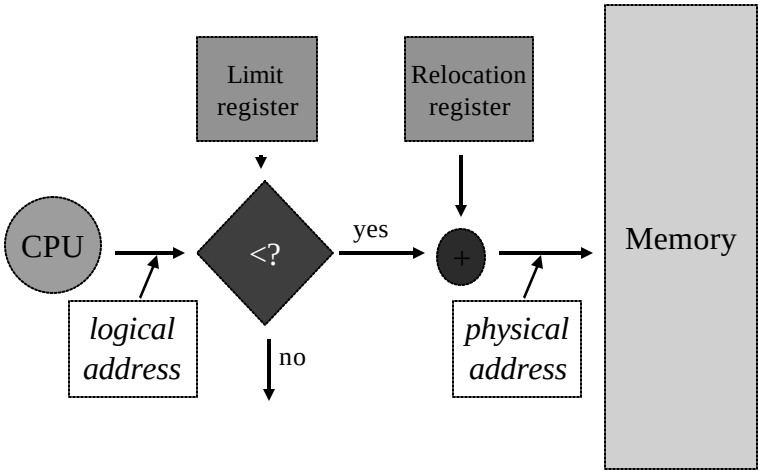
Single-Partition Allocation



Single-Partition Allocation With Transient Operating-System Code



Hardware Support For Limit and Relocation Registers



Multiple-Partition Allocation

- More than one user process can reside in memory at a time in contiguous memory allocations. There are two possible approaches:
  - There are a fixed maximum number of processes than can be resident at a time.
  - The number of process that can reside in memory can vary.

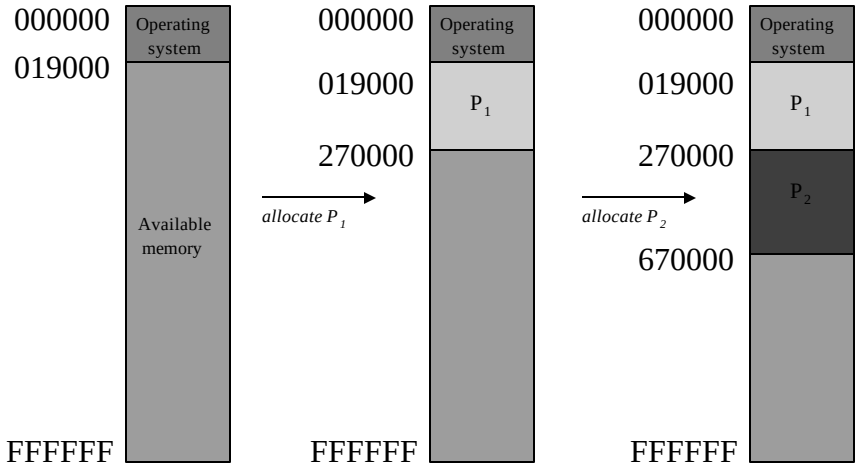
MFT

- MFT (***M***ultiprogramming with ***F***ixed number of ***T***asks) assumes that there will be up to N processes resident in memory.
- Memory is divided into N partitions of fixed sizes (that are not necessarily equal).
- MFT was used in IBM's OS/360 and is no longer used.

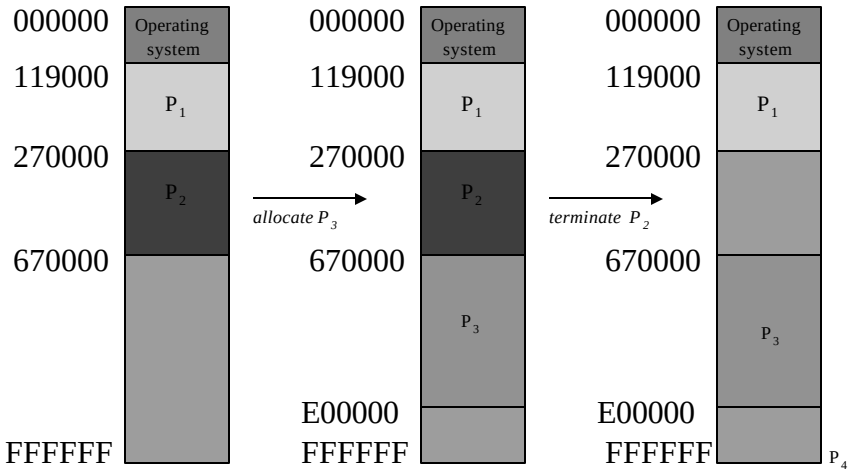
MVT

- MVT (***M***ultiprogramming with ***V***ariable number of ***T***asks) assumes that there is no fixed maximum number of resident processes.
- The operating system keeps track of what parts of memory are in use and when a process needs a memory allocation, it looks for a partition in memory in which it will fit.

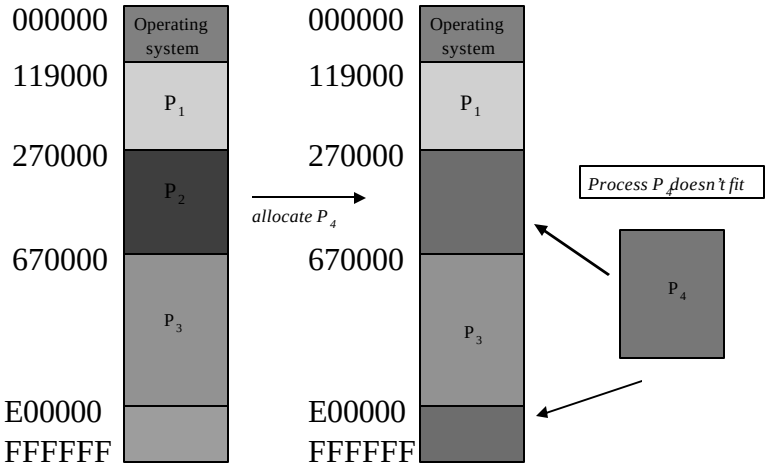
MVT and Allocating Memory



MVT and Terminating Processes



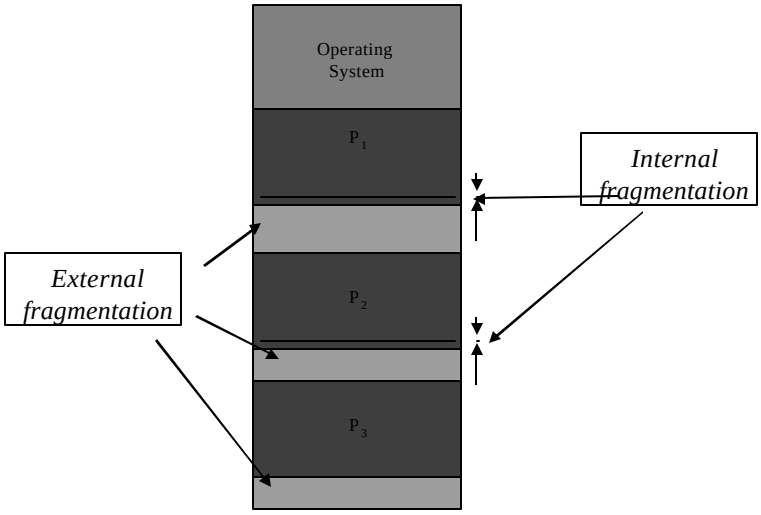
MVT and Fragmentation



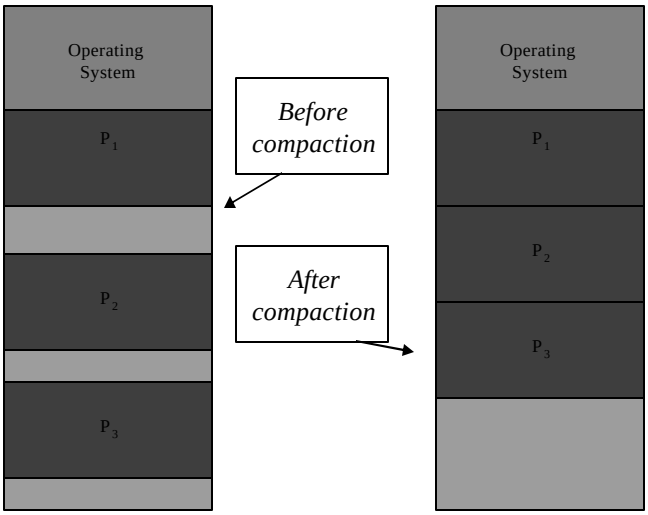
The Dynamic Storage-Allocation Problem

- There are three different algorithms that we can use to select an available block of memory into which the operating system will load the program:
- First fit
  - Best fit
  - Worst fit

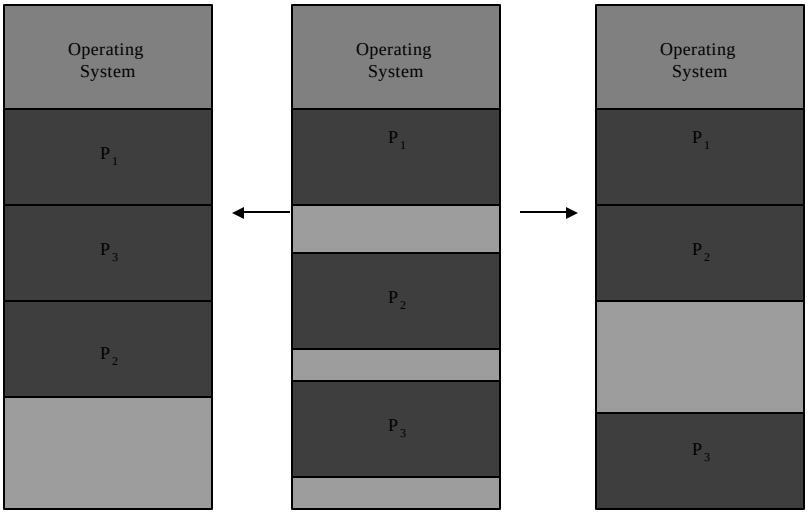
External & Internal Fragmentation



Compaction



Other Compaction Strategies



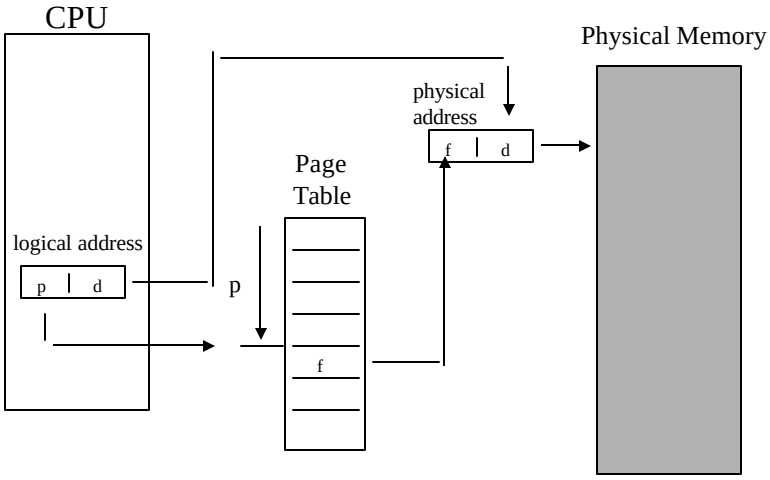
What Is Paging?

- In a paged memory management system, physical memory is divided into fixed-sized block called frames into which are loaded blocks of logical memory.
- All the addresses appear to be contiguous but they are not necessarily stored in contiguous areas of memory.
- This eliminates the problem of external fragmentation.

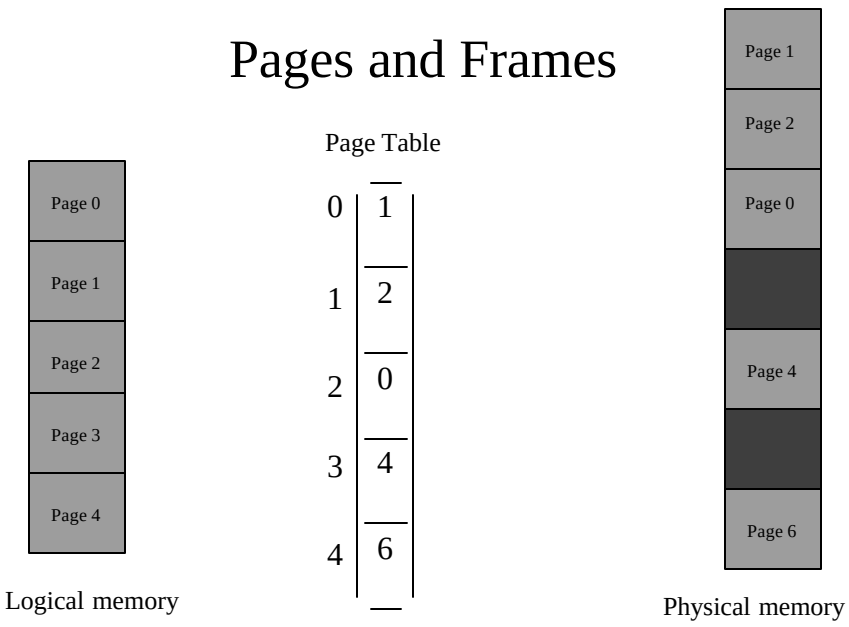
Logical and Physical Addressing in Paged Memory System

Logical Address	Page Number	Displacement
Physical Address	Frame Number	Displacement

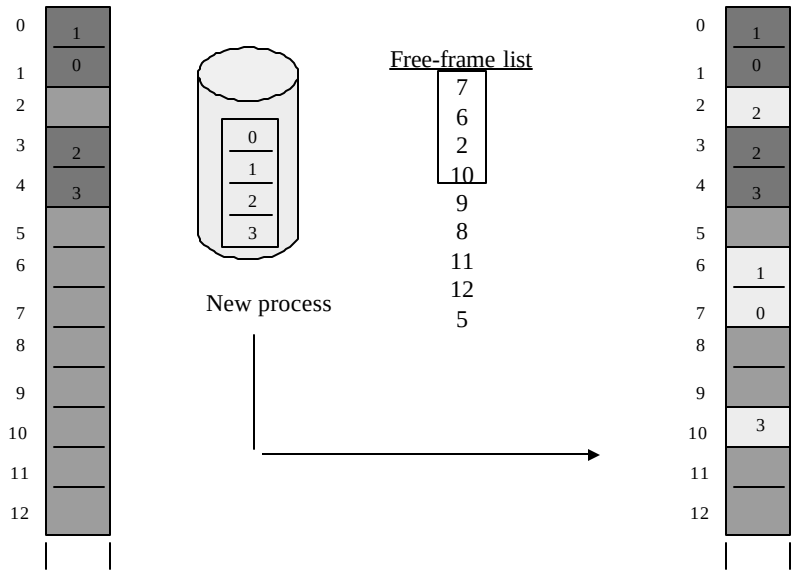
Hardware Support For Paging



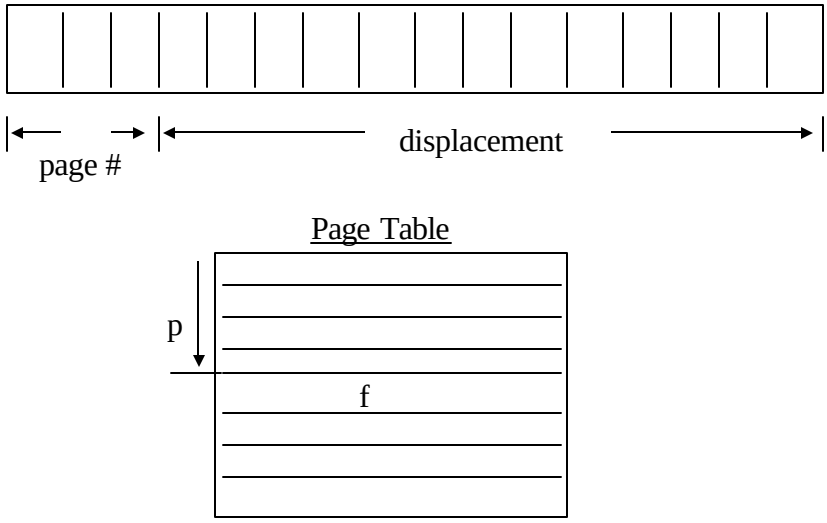
Pages and Frames



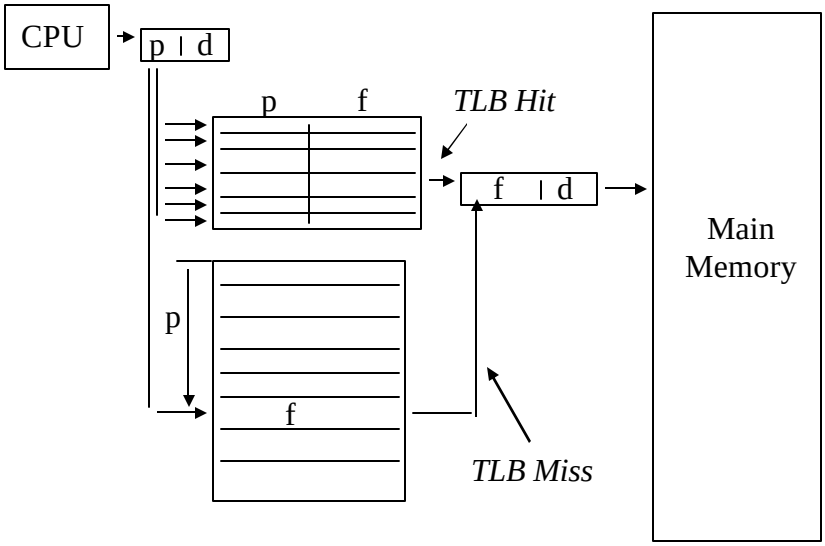
Loading a New Process



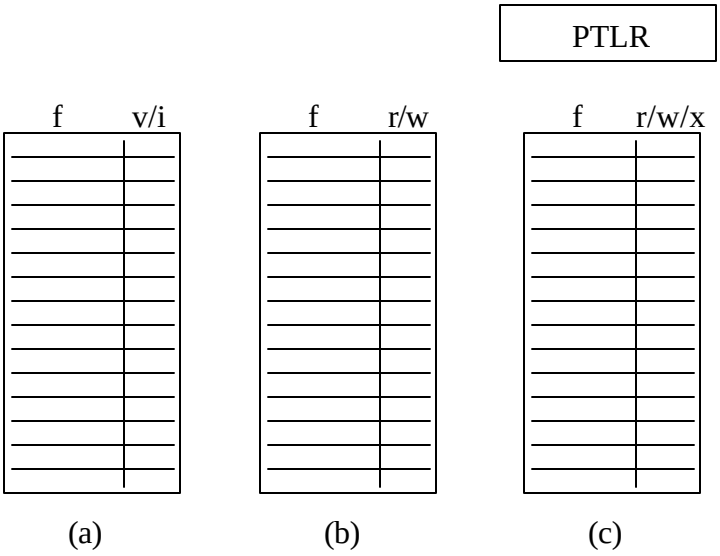
Using Registers For the Page Table



Associative Registers and Paging

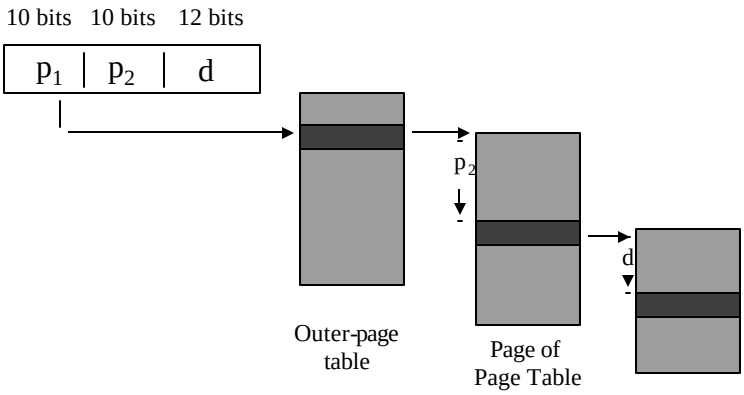


Paging And Memory Protection



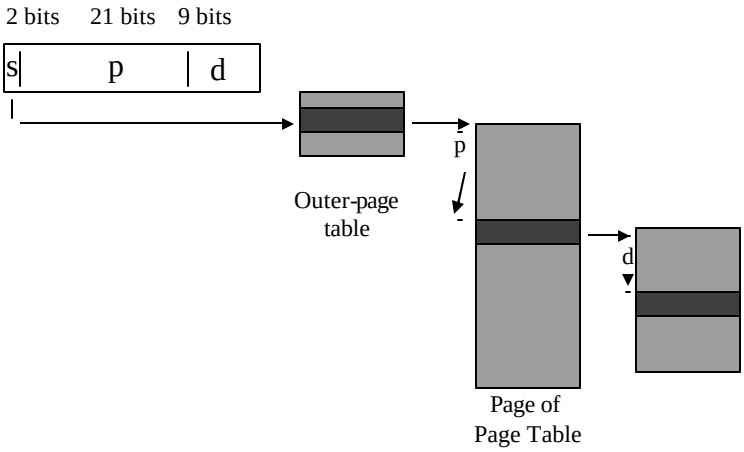
2-Level 32-bit Paging

<u>logical address</u>

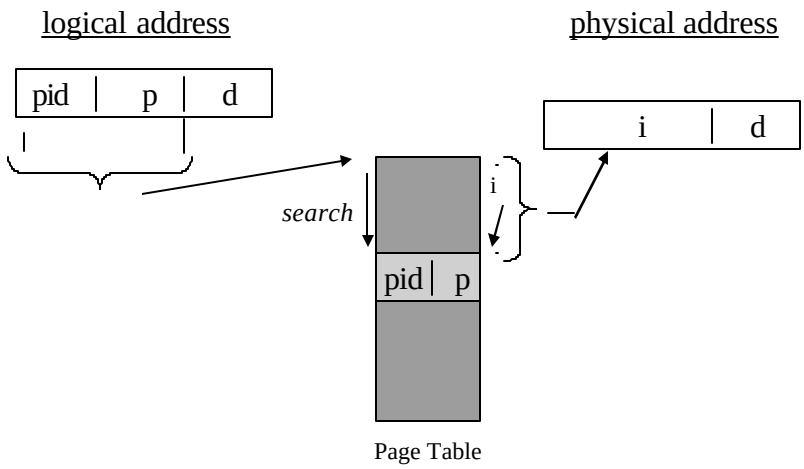


Paging on the VAX

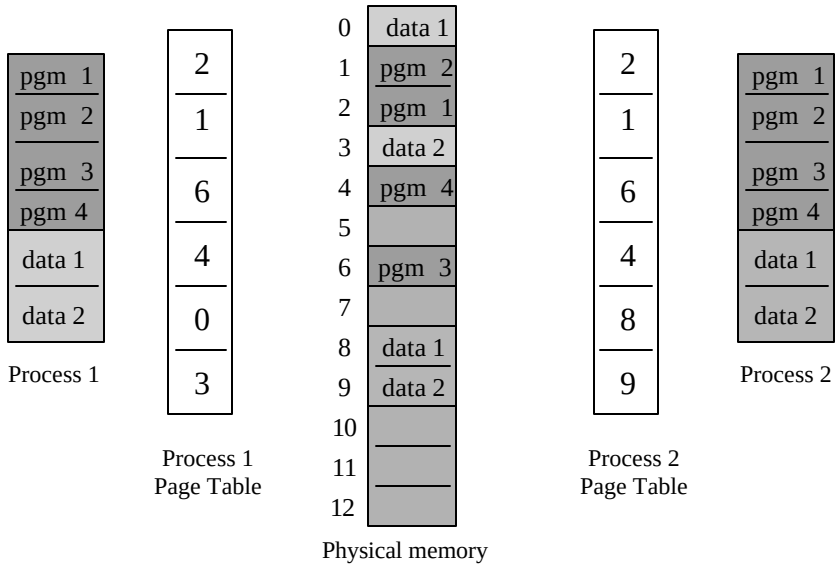
<u>logical address</u>



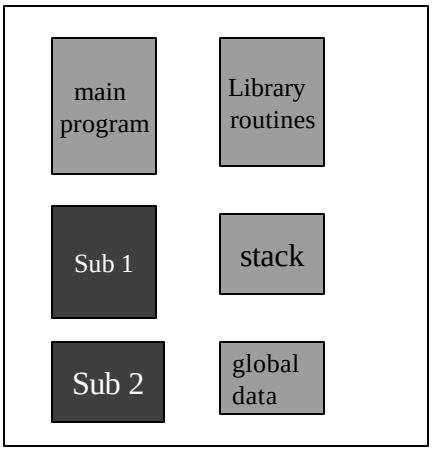
Inverted Page Tables



Shared Pages

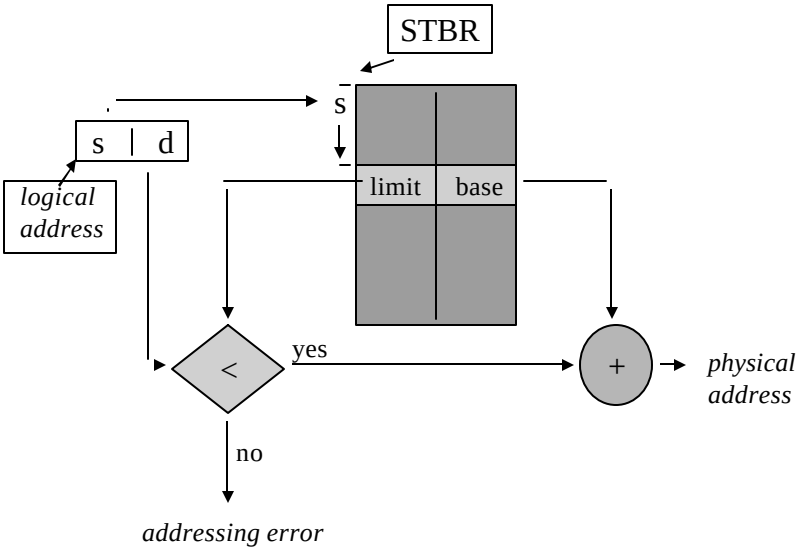


Segmentation



A user's program can be thought of as a collection of separate segments

Segmentation Hardware



Segmentation and Memory Protection

- Segments represent different components of a program, such as code and data.
  - Code segments can be marked as read-only or execute-only.
  - Segments containing arrays can check array indices automatically.
  - Code and data segments can be shared by different processes.

Segments and Fragmentation

- Memory allocation works in a manner similar to paging, except that segments are of variable size.
  - There will external fragmentation, which can be alleviated by compaction.
  - The problem can be minimized by working with smaller segments.

Segmentation With Paging

- The two approaches could be used together to improve memory management
  - MULTICS paged the memory segments (which could be as large as 64K) to minimize external fragmentation.
  - The 32-bit version of OS/2 took a similar approach.