Memory Management

H. Levy

Address Binding

- **Programs share memory and are executed in memory, because instructions refer directly to memory addresses.**
- **Your program refers to its data symbolically, but at some point a variable must be assigned an address; we call this *binding*. It can be done at various times:**
  - compile time: compiler could assign addresses if it knows where code will go. Problem: separate module compilation.
  - link time: a linker prog. can take several (relocatable) modules and produces a linked image.
  - load time: when the prog is loaded, addresses can be bound
  - execution time: sometimes addresses can be changed at run time.

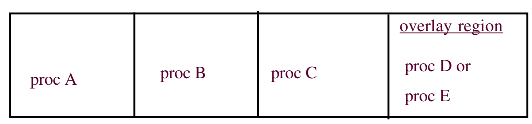
Loading and Linking

- **Dynamic Loading: procedures are kept on disk in relocatable form. a proc is loaded when it's first called.**
- **Dynamic Linking:**
  - program contains "stubs" that know how to find each procedure and load it.
  - on first call, stub loads and links routine, and replaces itself with the routine address
  - on subsequent calls, the routine is called directly
  - a single system routine might be shared in memory by several application programs
  - uses OS support

Overlays

- **Overlays: an old technique used mainly on 16-bit minicomputers without VM.**
  - explicit calls in the program are made to load data and code into memory
  - if that data is not needed, it is later replaced with other code and data (i.e., the addresses are reused)
  - this is an incredibly pain!

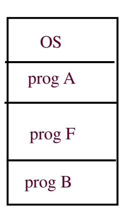
procs D and E don't have to execute at the same time, so they can be overlayed



address space

Memory Allocation

- **There are various schemes for allocating memory to programs and the OS.**
  - nothing: 1 program and OS reside in memory
  - multiple partitions: divide memory up into sections used for each program (requires protection)



Allocation

- **Problem with partition scheme is *external fragmentation*.**
- **As jobs finish they leave various sized holes in memory.**
- **Schemes for putting new jobs into memory holes:**
  - first fit: use first hole big enough
  - best fit: find smallest hole big enough
  - worst fit: find largest hole, leaving largest remainder
- **first fit and best fit do better. A sorted list helps performance.**
- **after a while, fragmentation can waste significant space, requiring compaction by the OS.**

Other Solutions

- A better way to manage memory may involve paging or segmentation or other hardware-supported reloaction schemes.

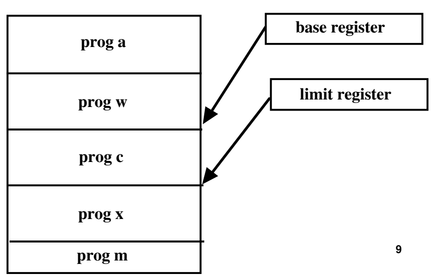
Logical vs. Physical Addresses

- Programs use *logical* addresses (*virtual* addresses) to name memory locations.
- These logical addresses are independent of the actual physical addresses at which the program resides.
- Where a program exists in memory depends on the OS.
- As a program runs, it issues logical addresses, which are translated to physical addresses by the hardware
- The set of all addresses issued by a program (or namable by a program) is called its logical (virtual) address space.

Simple Relocation

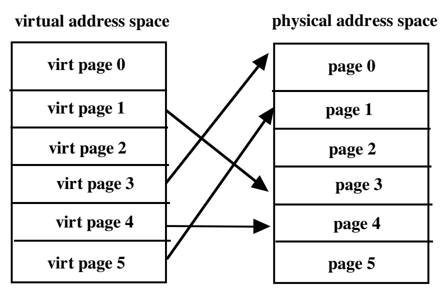
- Earliest method of supporting program relocation was base/limit registers.

when a program is running, the base and limit registers define its boundaries



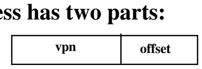
Paging

- Paging solves the external fragmentation problem by using fixed sized units in both physical and virtual memory.

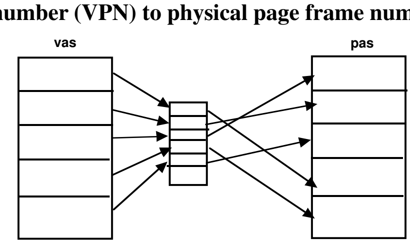


Paging

- In a paged system, each virtual address has two parts: virtual page number and offset.



- Page size is determined by hardware.
- A Page Table contains the mappings from virtual page number (VPN) to physical page frame number (PFN).



Paging

- User view memory as one contiguous address space from 0 through n.
- In reality, pages are scattered throughout physical storage.
- The mapping is invisible to the program; it is maintained by the OS and used by the hardware on each reference by the program.
- Protection is provided because a program cannot reference memory outside of its VAS.
- Hardware always contains a TLB to speedup the page table lookups.

Sharing

- Paging introduces the possibility of sharing memory.
- Several processes can share one or more pages, possibly with different Protection.
- A shared page may exist in different parts of the VAS of each process. (This is difficult if it contains internal addresses.)

Segmentation

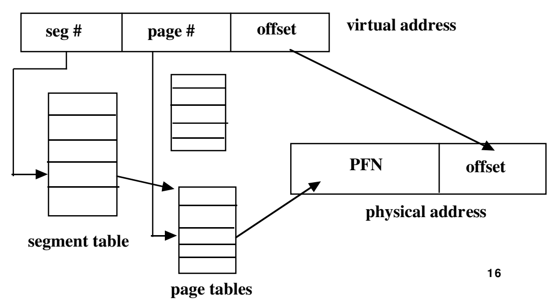
- Segmentation relates more to the programmer's view of memory as containing a set of program objects or structures.
- Each object or structure is allocated a segment of memory.
- A virtual address space consists of a collection of segments.
- Each word is addressed by segment number plus offset.
- Segments can independently grow and shrink dynamically.

Segmentation

- Segmentation is supported by a segmentation table, which is like multiple base/limit registers.
- Each segment table entry contains: the base address in primary memory of that segment, the length of that segment, and some protection information.
- Segmentation is better basis for sharing than paging. Programs share segments -- logical units.

Segmentation and Paging

- It's possible to combine the advantages of the two schemes.



Segmentation and Paging

- Combination allows programs to deal with segments (logical units) and to share segments.
- Segments are paged for better OS storage management.
- Page management is simplified in the face of sharing because there is only one PTE for each page (in a pure paging system, a shared page would have multiple PTEs and multiple valid bits, etc)
- Segmentation provides multiple directions of dynamic growth.