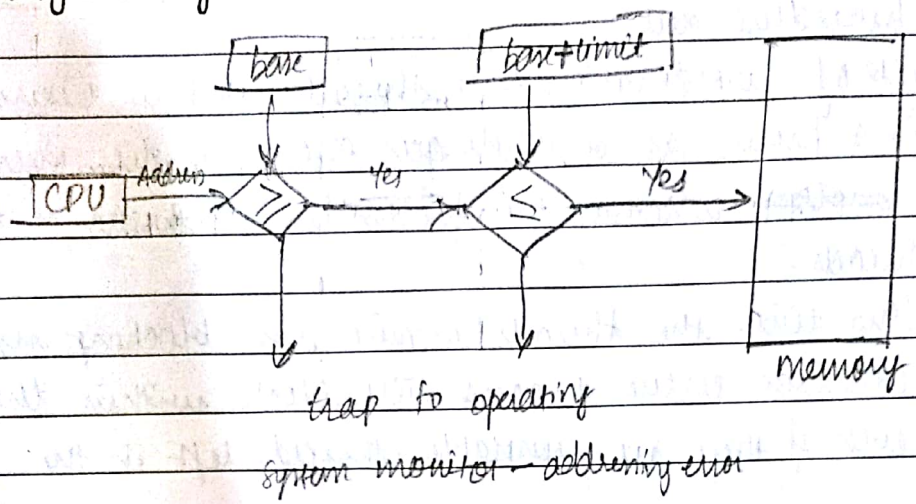
Lecture-20
BDU

classmate
Date 19/03/18
Page

Memory management



Binding of instructions and data to memory

~~Address~~ binding of instruction and data to memory addresses can happen at three stages -

- Compile time : If memory loc^n is known a priori, absolute code can be generated (recompile if starting addr. changes)
- Load time : Must generate relocatable code if memory location is not known at compile time.
- Execution time : Binding delayed until run time if process can be moved during its execution from one memory segment to another.
  - Need h/w support for address maps (e.g., base and limit registers)

(static binding)

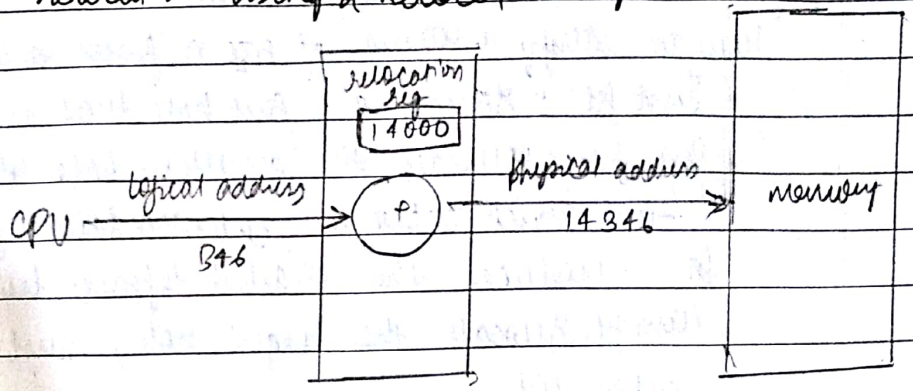
Logical vs. Physical address space

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 same in compile-time & load-time address-binding schemes ; they differ in execution-time binding.

Dynamic relocation using a relocation register



- Routine is not loaded until it is called.
- Better memory utilization; unused routine is never loaded.
- All routines kept on disk in relocatable load format.

Swapping (done by dispatcher)

- 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 accomodate 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.
- System maintains a ready queue to ready-to-run processes which have memory images on disk.

Contiguous allocation

- Main memory must support both OS and user processes
- Limited resource, must allocate efficiently
- Contiguous allocation is one early method
- Main memory usually into two partitions
  - Resident OS, usually held in low memory with interrupt vector
  - User processes then held in high memory
  - Each process contained in single contiguous section.

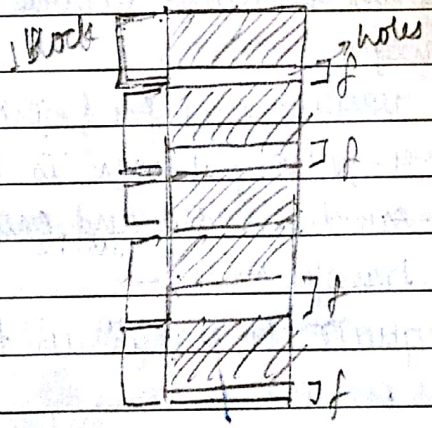
Dynamic storage allocation problem

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

- First fit : Allocate the first hole that is big enough
- Best fit : Allocate the smallest hole that is big enough; must search entire list, unless ordered by size
  - Produces the smallest leftover hole
- Worst fit: Allocate the largest hole, must also search entire list.
  - Produces the largest leftover hole

Fragmentation

- External fragmentation – Total memory space exists to satisfy a request, but it is not contiguous.
- Internal fragmentation – Allocated memory may be slightly larger than requested memory; this size difference is memory internal to a partition, but not being used.
- First fit analysis reveals that given N blocks allocated, 0.5N blocks lost to fragmentation.
  - 1/3 may be unusable → 50% rule



- Reduce external fragmentation by compaction
  - Shuffle memory contents to place all free memory together in one large block.
  - Compaction is possible only if relocation is dynamic, and is done at execution time.
  - I/O problem
    - Latch job in memory while it is involved in I/O.
    - Do I/O only into OS buffers.

classmate
Date
Page

- Now consider the backing store has same fragmentation problems-

Compaction → Shuffling required
Linking mechanism → Shuffling not required.

Lecture-21
B2w

20/03/18.

Overlays - Frequently used instructions are in the primary memory but the rest are kept in secondary storage. And when needed they are swapped into memory.