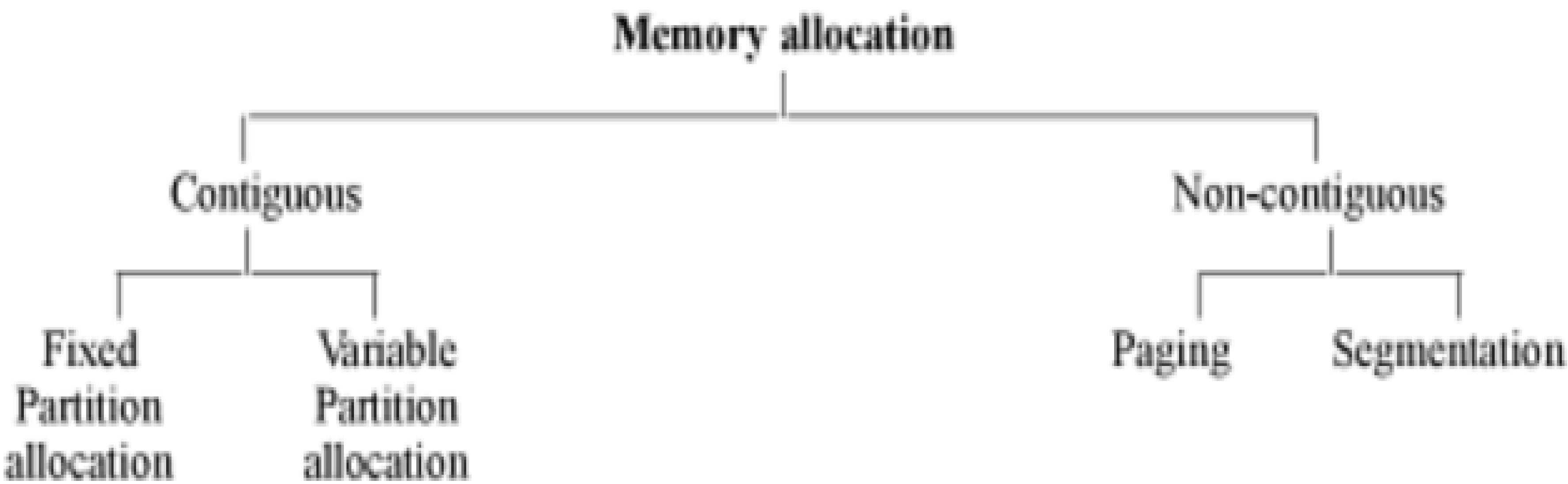
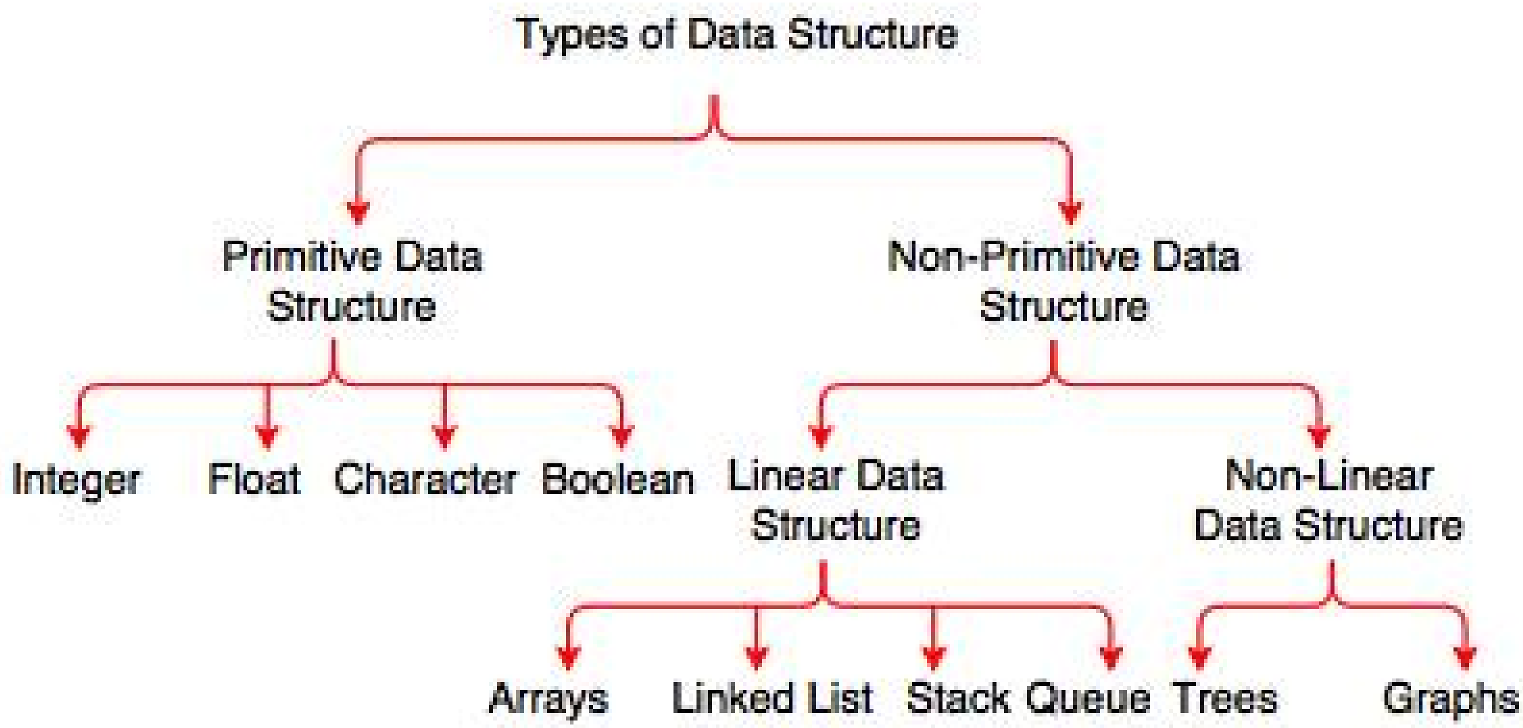
Memory management techniques

- Various memory management techniques exist such as partitioning memory, dynamic memory allocation, overlays, swapping. But the two most important and widely used strategies are paging and segmentation.



Example:

- Data structure is an arrangement of data in computer's memory.



Partitioning

- In this method, the OS manages the main memory by dividing the memory into regions of fixed or variable.

- The two types of partitioning are:
  - Static/Fix-Sized Partitions
    - Equal-size
    - UnEqual-size
  - Dynamic/Variable-Sized Partitions

Fixed-size partitions; Equal-size

- Memory is divided into *n* fixed-sized partitions in which different processes can be loaded.
- Any process whose size is less than or equal to the partition size can be loaded into an available partition
- If all the partitions are full, the operating system can swap a process out of a partition.

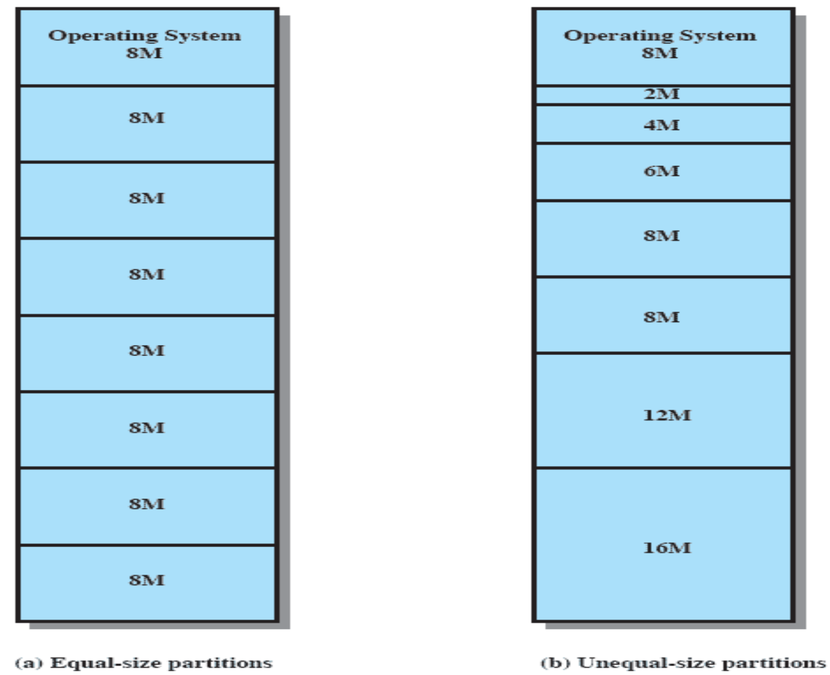
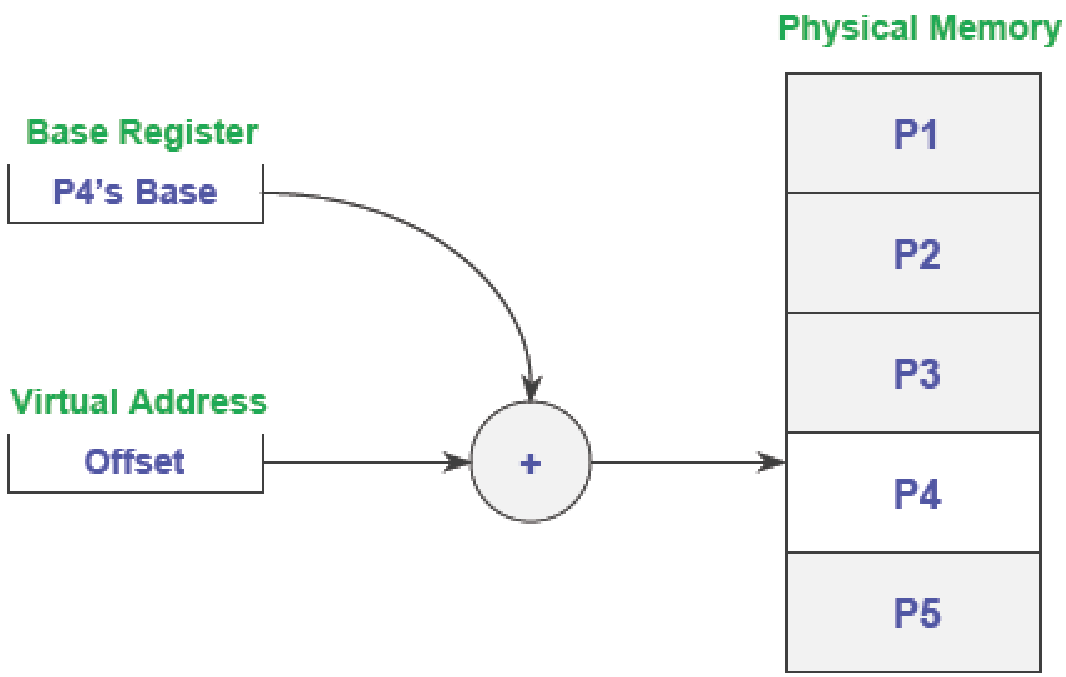


Figure 7.2 Example of Fixed Partitioning of a 64-Mbyte Memory

Placement Algorithm

- If there is an available partition, a process can be loaded into that partition because all partitions are of equal size, it does not matter which partition is used.
- The hardware requirements are base and limit registers. Physical address = virtual address + base register. The Base register is loaded by OS when it switches to a process.



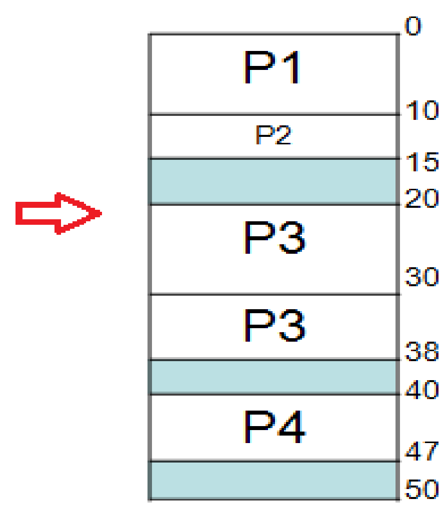
Problem

- Any process whose size is less than or equal to the partition size can be loaded into an available partition.
  - One program may be **too large** to fit in one partition. This means that the program needs to be designed with the use of overlays. With this, main memory use is inefficient.
  - One program may be **too small** to occupy the entire partition. The space wasted inside of allocated memory blocks is called internal memory fragmentation. This space is unavailable for use by the system until that process is finished and the area is released.

Static Partition

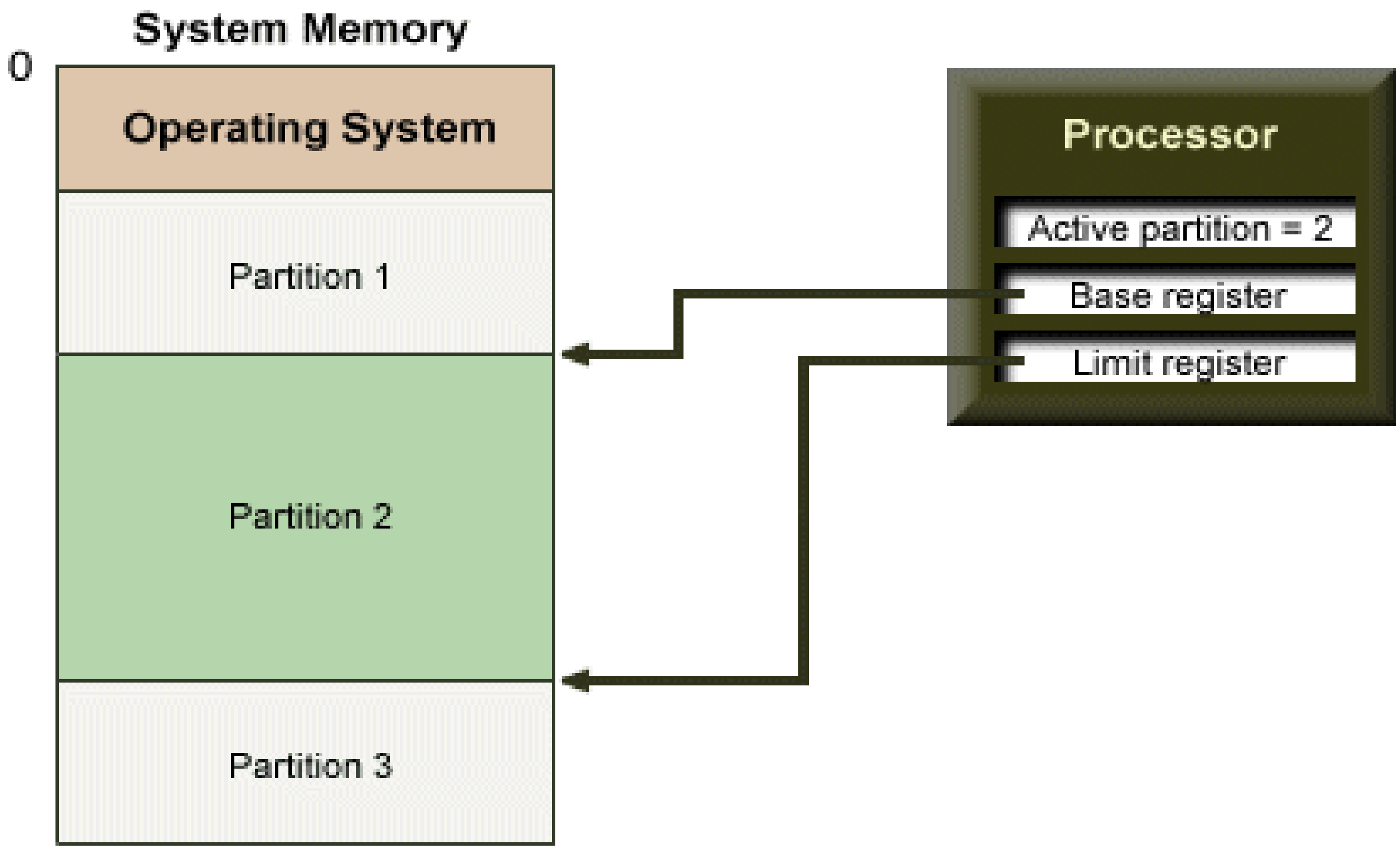
process	space
P1	10
P2	5
P3	18
P4	7

Partition size=10



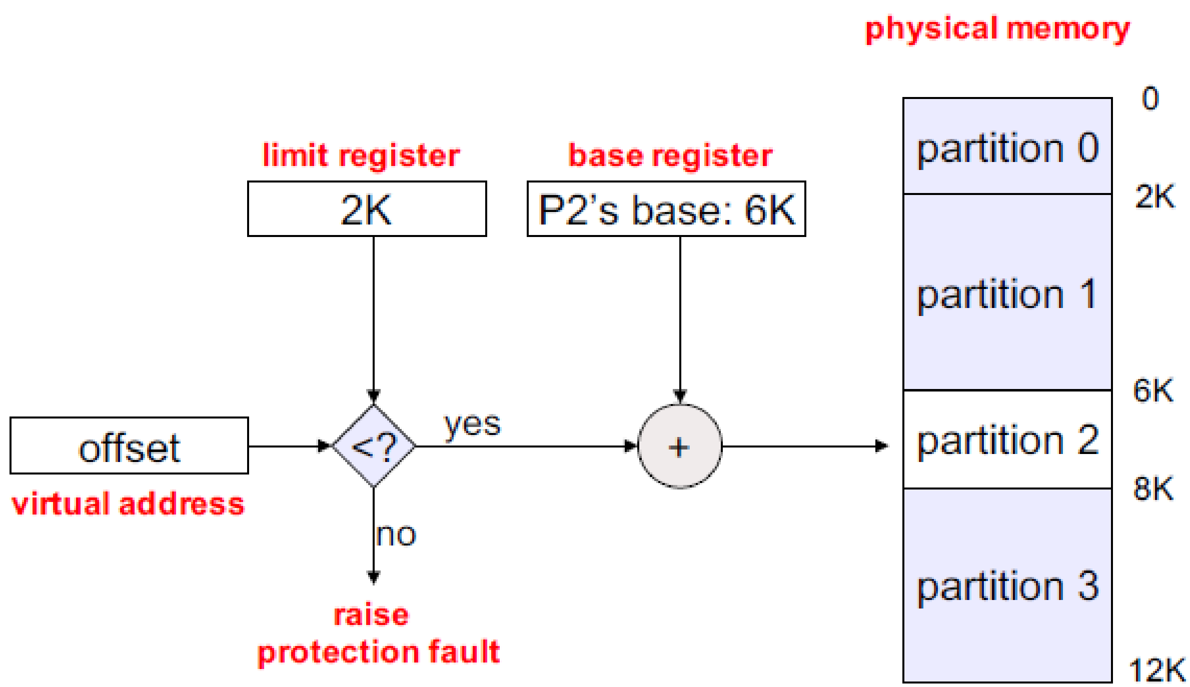
Fixed-size partitions; Unequal-size

- With unequal partitions, the internal fragment problem is reduced, but not eliminated.
- The hardware requirements: base register + limit register.



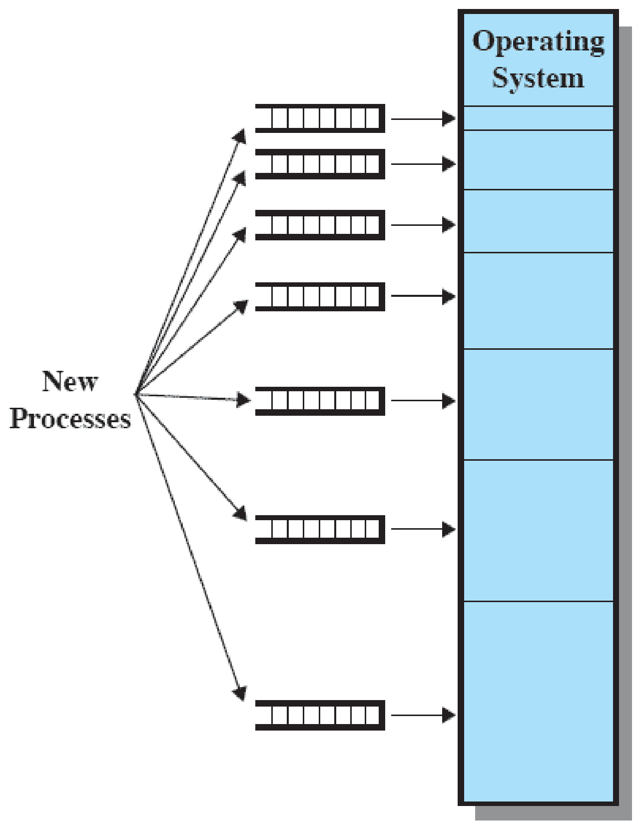
Fixed-size partitions; Unequal-size cont…

Mechanics of fixed partitions

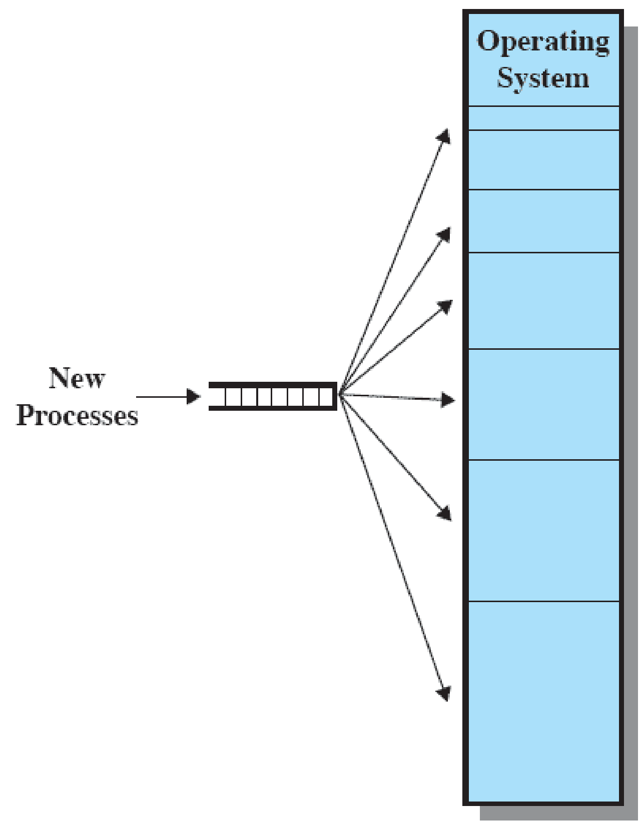


Placement Algorithm

- This model can be implemented in two ways:
  - multiple queues: there is one queue for each partition
  - single queue: there is one single queue for all partitions



(a) One process queue per partition



(b) Single queue

multiple queues

- In multiple queues, it assigns each process to the smallest partition within which it will fit.

- The advantage is that it decreases internal fragment.
- However, the problem is that some queues will be empty if no processes within a size range is present.

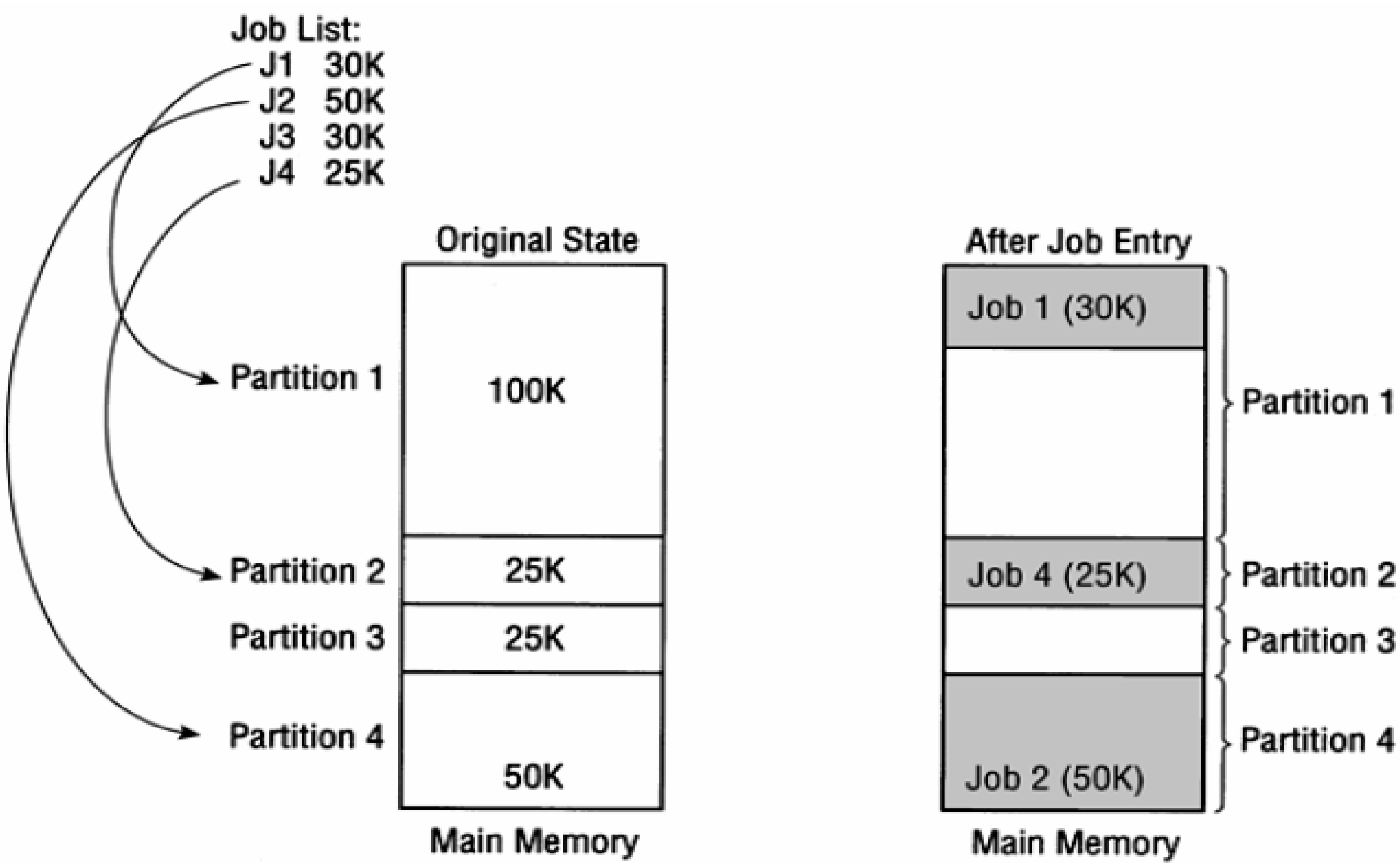
single queues

- In single queue, when its time to load a process into main memory the first available partition within which it will fit is selected.

- It increases the level of multiprogramming at the expense of internal fragmentation

single queues cont…

- Job1 is allocated first space large enough to accommodate it. Job 2 is allocated next available space large enough to accommodate it. Job 3? No partition is big enough to accommodate it.



Fixed-size partitions cont…

- As we saw, in Fixed Partitioning method with either equal or unequal sizes:
  - any process whose size is less than or equal to a partition size can be loaded into the partition.
  - if all partitions are occupied, the operating system can swap a process out of a partition if no process is in the Ready or Running state.
  - In case, when a program is too large to fit in a partition, the programmer must then design the program with overlays.

Overlay

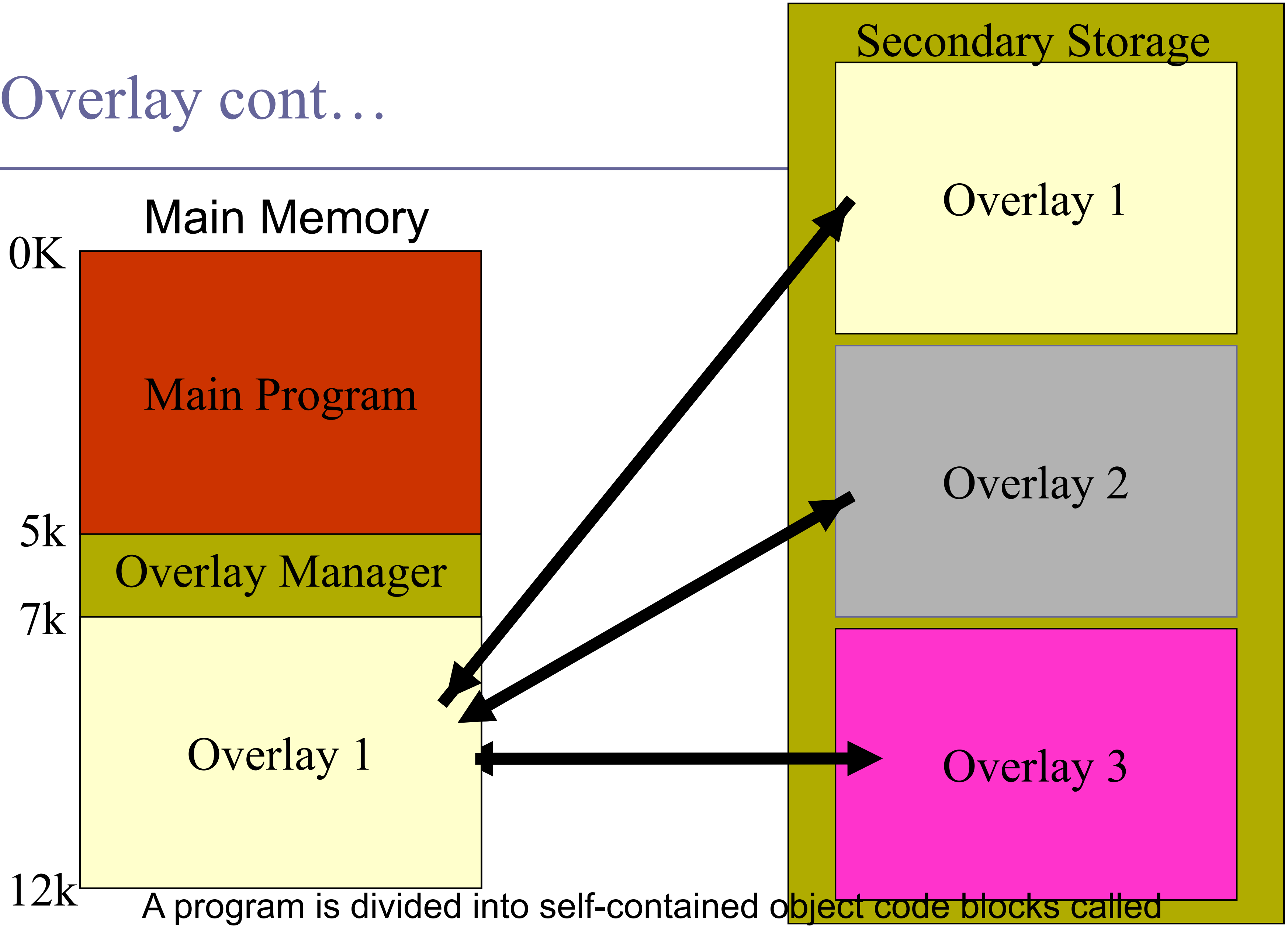
- In cases when a process is larger than the amount of memory, a technique called overlay can be used.
- The programmer breaks the code into pieces, called overlays, that fit into RAM.
- An Overlay manager/driver loads an overlay segment when it is not in RAM.

Overlay cont…

- This method keeps in memory only those instructions and data that are needed at any given time. When other instructions are needed, they are loaded into space by Overlay driver/manager that was occupied previously by instructions that are no longer needed. It means the same memory is **overwritten/overlay** for running different phases of the program.

- While no special support needed from OS, the programming design of overlay structure is complex.

Overlay cont…



A program is divided into self-contained object code blocks called overlays where the size of the overlay is limited according to memory constraints.

Example:

- Consider a two-phases code:
  - Pass1 constructs a symbol table.
  - Pass2 generates machine-language code.

	Size (k = 1024 bytes)
Pass1	70k
Pass2	80k
Symbol table	20k
Common routines	30k
Total size	*200k*

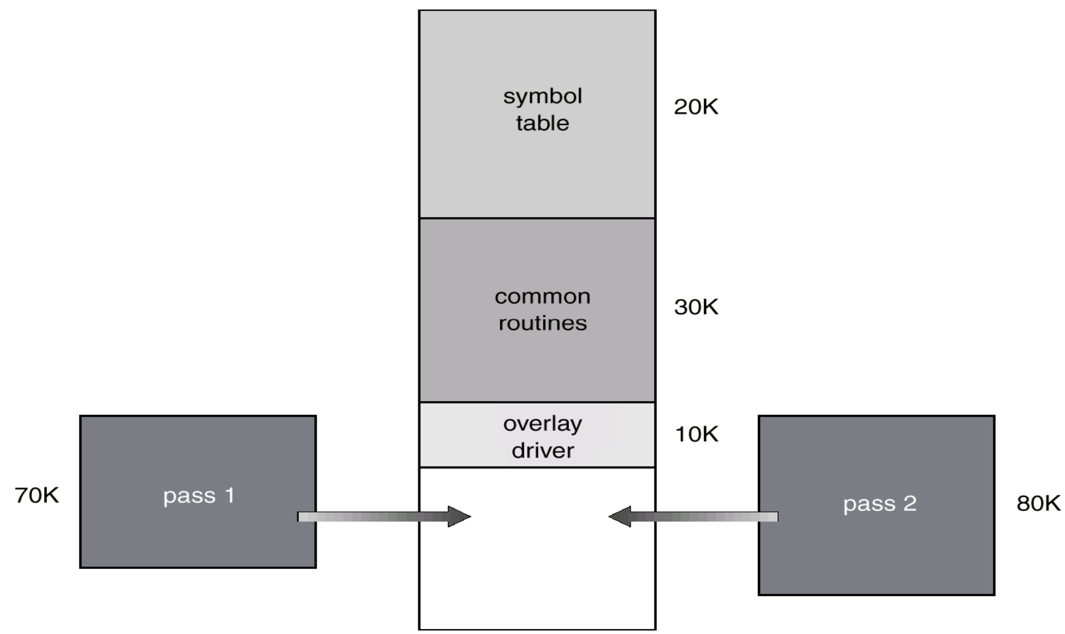
- To load everything at once, we need 200k of memory. Thus if only 150K is available, we cannot run our process.

Program symbols: names of variables and functions
Symbol table: name and current location of variables or functions

Example cont…

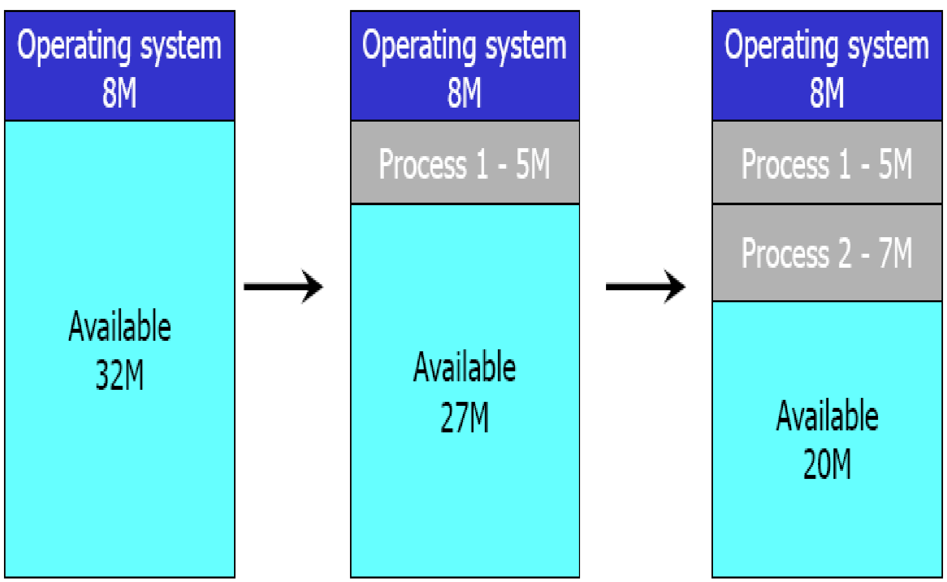
- Phase1 and Phase2 do not need to be in memory at same time. So, we define two overlays:
  - Overlay A: symbol table, common routines, and Pass1.
  - Overlay B: symbol table, common routines, and Pass2.
- We add overlay driver 10k and start with overlay A in memory. When finish Pass1, we jump to overlay driver, which reads overlay B into memory **overwriting** overlay A and transfer control to Pass2.

	Size (k = 1024 bytes)
Pass1	70k
Pass2	80k
Symbol table	20k
Common routines	30k
Total size	*200k*



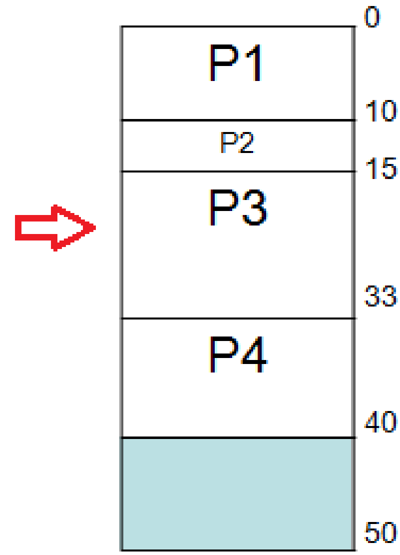
Variable-sized partitions

- The memory partitions are created dynamically in response to process requests. When a process is brought into memory, a partition of exactly the right size is created to hold it, eliminating internal fragmentation.
- Hardware requirements are base register and limit register.



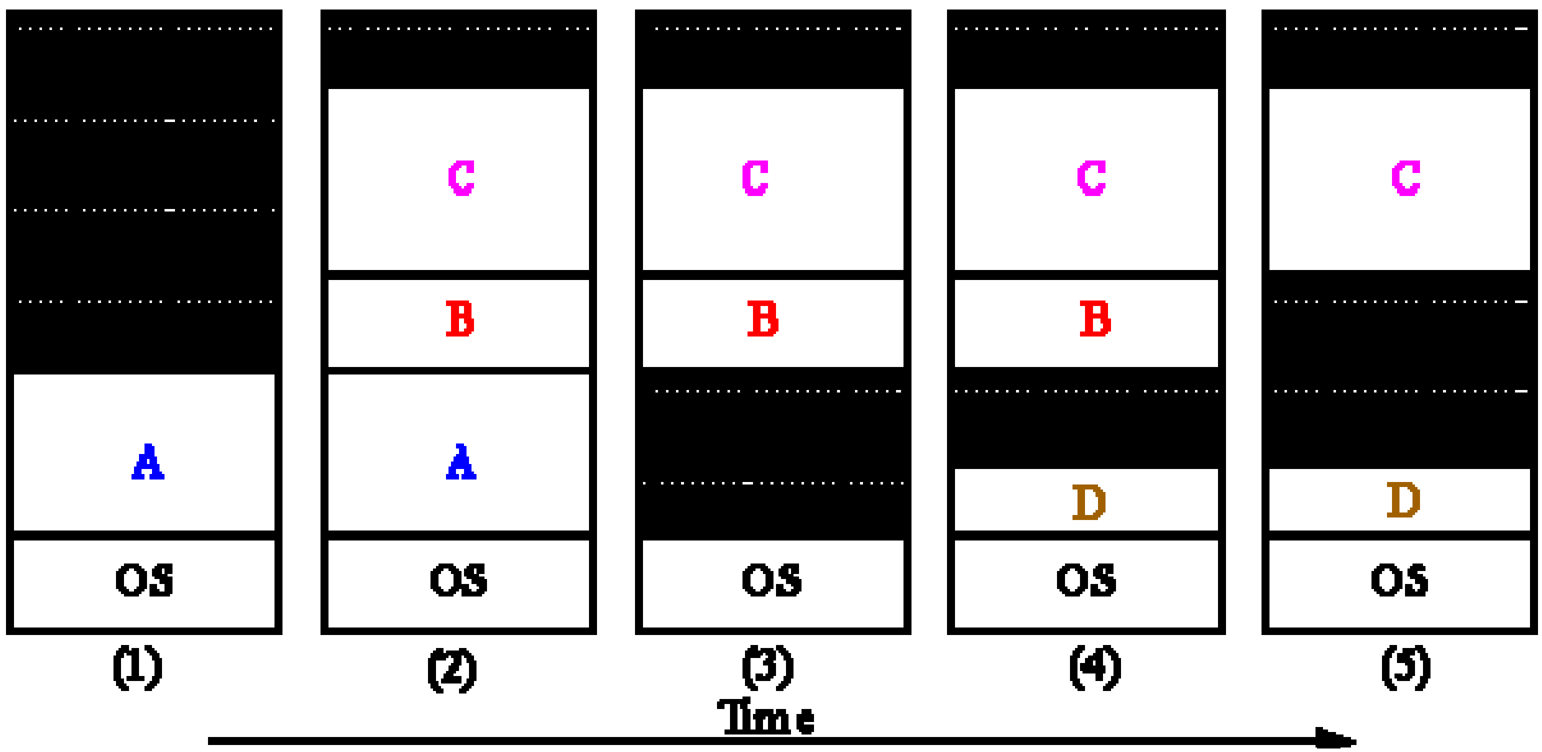
Dynamic Partition

process	space
P1	10
P2	5
P3	18
P4	7



Variable-sized partitions cont…

- For example, initially, process A is in memory, then B and C are created. A terminates, D is created, B terminates.

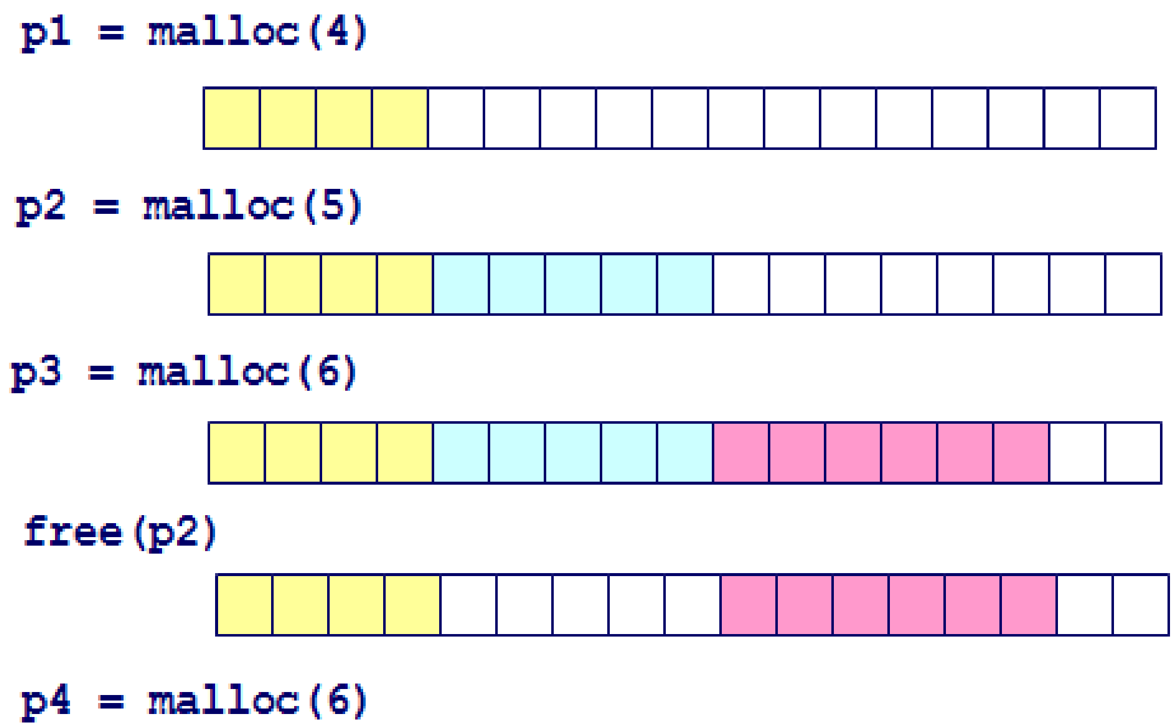


Placement Algorithm

- One obvious question suggested by dynamic partitioning is “Where do we place a new process?” Three simple algorithms exist :
  - First fit: Scans memory form the beginning and chooses the first available block that is large enough
  - Best fit: Allocate the smallest block that is big enough; must search entire list, unless ordered by size. Produces the smallest leftover hole.
  - Worst/Next fit: Allocate the largest block; must also search entire list. Produces the largest leftover hole.

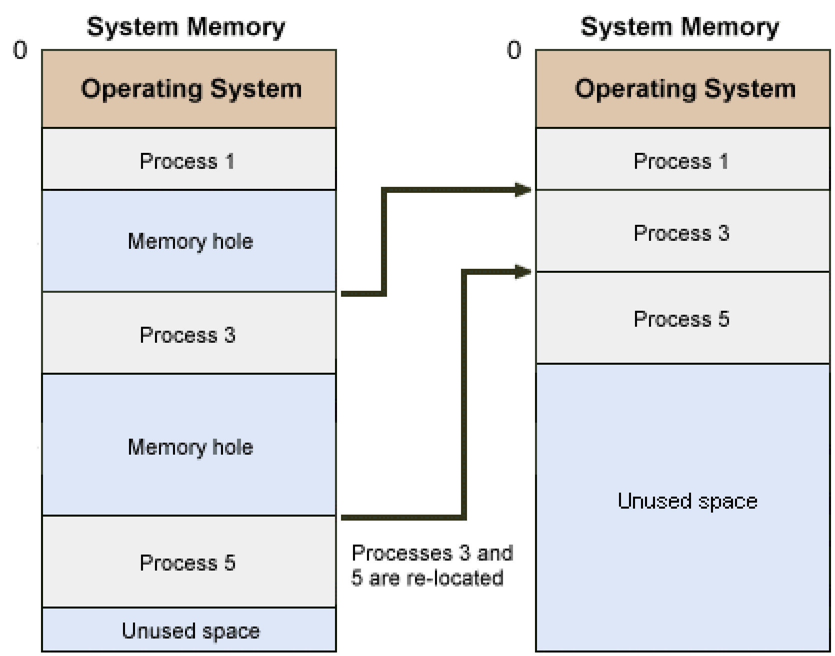
Problem

- Dynamic partition size improves memory utilization but complicates allocation and de-allocation by creating holes. Holes are areas of unused available areas of memory between partitions. They are usually too small to hold a process. The external memory fragmentation occurs when there is enough free memory for a process to run, but no single free block is large enough.



Problem cont…

- The solution is memory compaction which combines all the holes together so they are contiguous allowing a process to move into that space.
- Memory compaction has a large overhead (takes a considerable amount of time to complete) and it also requires processes to be relocatable. Therefore, compaction method is slow and costly and is rarely used.



Non-contiguous memory allocation

- As we saw, the problems with using contiguous memory allocation, are fragmentation and inefficiencies (algorithms to track holes and compaction).

- So, a better method to overcome the fragmentation problem is to make our logical address space non-contiguous.

Virtual memory

- Before virtual memory, programs that were too big for the size of physical memory used Overlay.
- However, overlay was difficult for programmer to manage.
- Then Virtual memory was developed.
- Virtual memory removes burden of memory resource management from the programmer.

Virtual memory cont…

- This method in fact is implementing memory on the hard drive. The hard disk can be used to allow more processes to run than would normally fit in main memory.

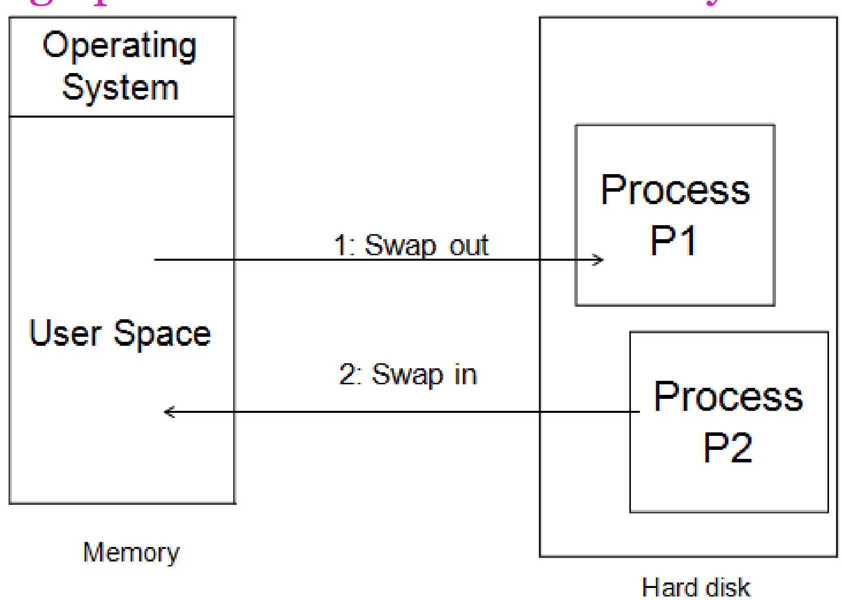
- By default, Windows sets the initial virtual memory paging file as equal to the amount of RAM and limits this increase to three times the amount of RAM to ensure system stability.

- One scheme is that when a process blocks for I/O (e.g. keyboard input), it can be swapped out to disk, allowing other processes to run.

- Virtual memory can be implemented today in 2 ways:
  - Swapping (old; an early virtual memory technique)
  - Paging
  - Segmentation

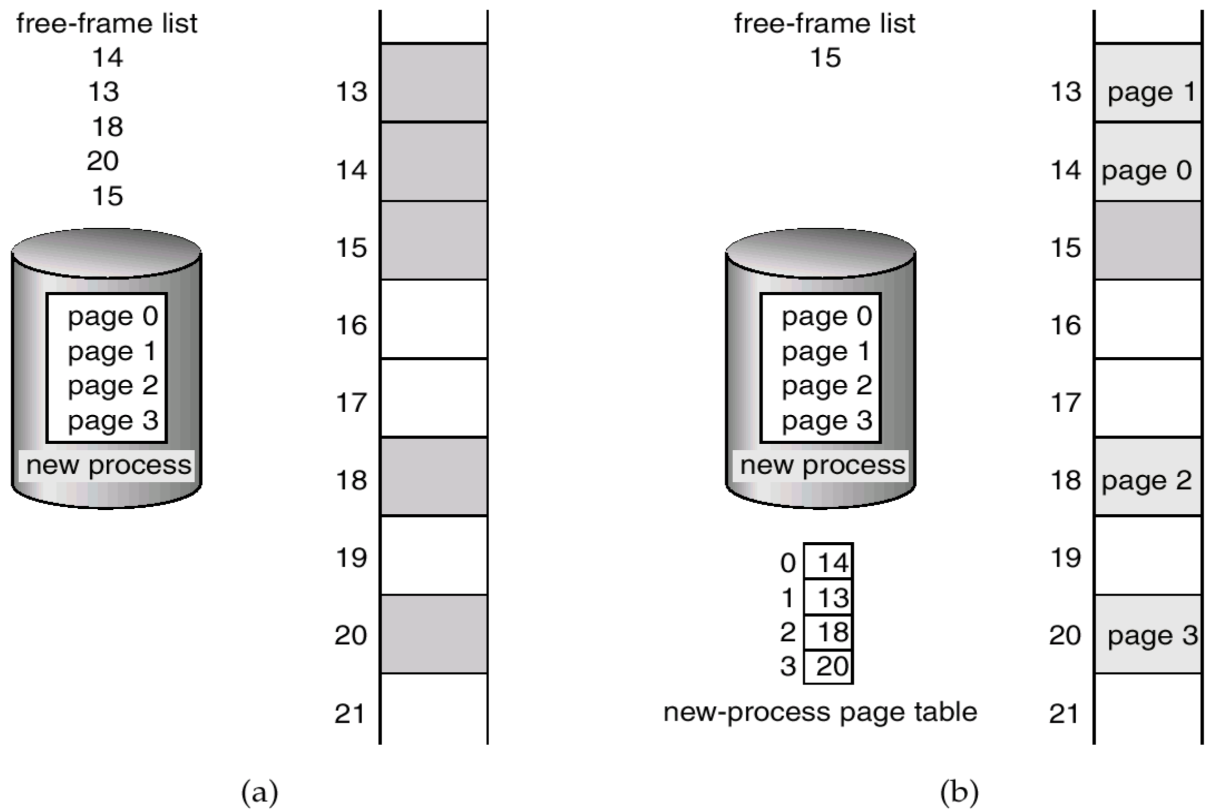
Swapping

- Swapping is the act of swapping the entire process from physical memory into the virtual memory portion of the hard drive.
- In time sharing and multiprogramming environment, there are many processes running simultaneously in the system. What if there is not enough memory to hold all the processes that want to run? The first solution that comes to mind is to move processes that have been suspended or preempted to disk to make room for other ones that want to run. When it is their turn to run again, they can be brought back into memory. The method of dynamically moving processes between memory and disk is called swapping.



Paging

- In paging, physical memory is divided into fixed size partitions called frames. Virtual address space of each process is also divided into blocks of same size called pages. Each page and its corresponding frame must have the same size. The corresponding pages of the process are loaded into any available memory frames.



Paging cont…

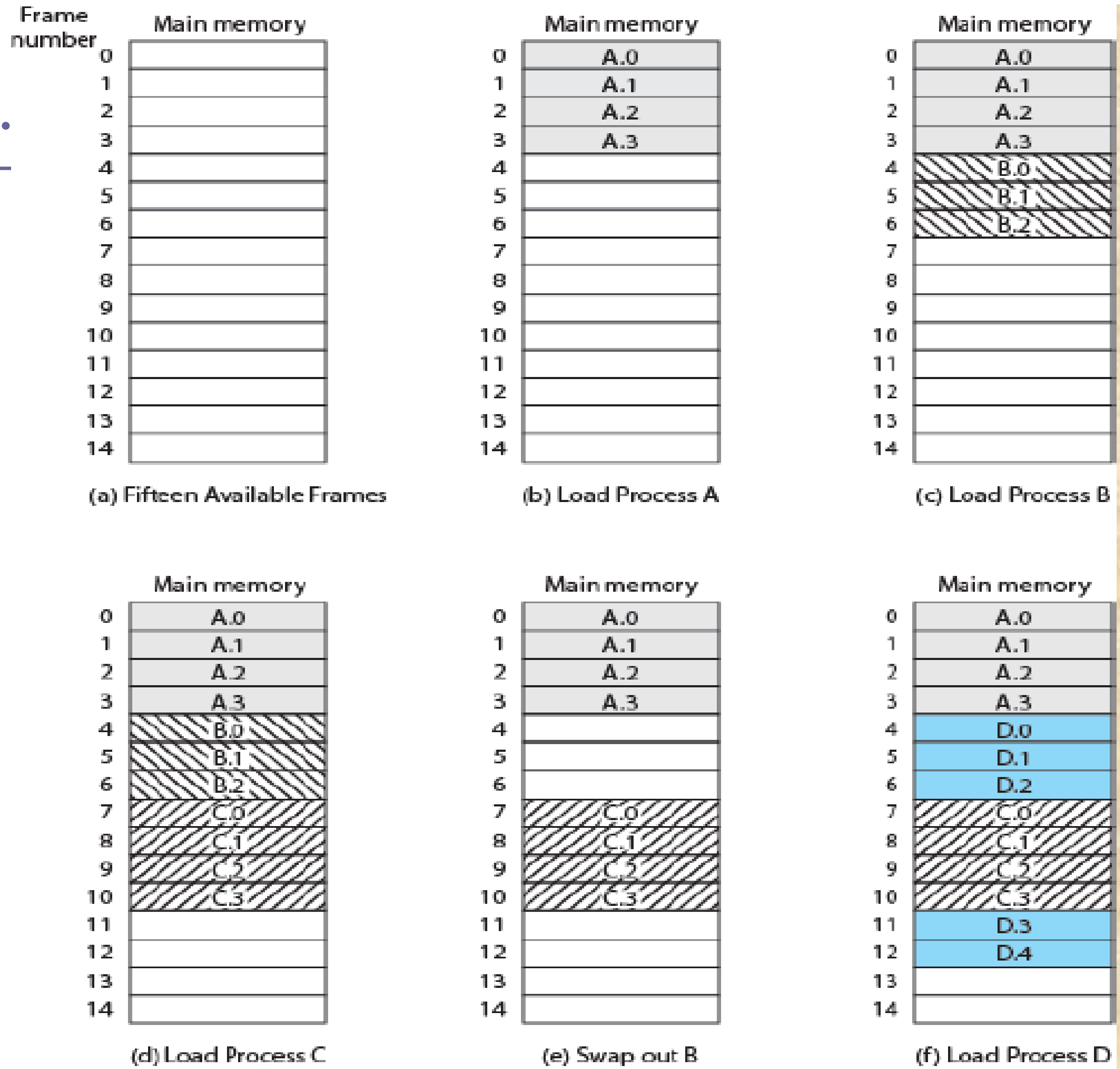
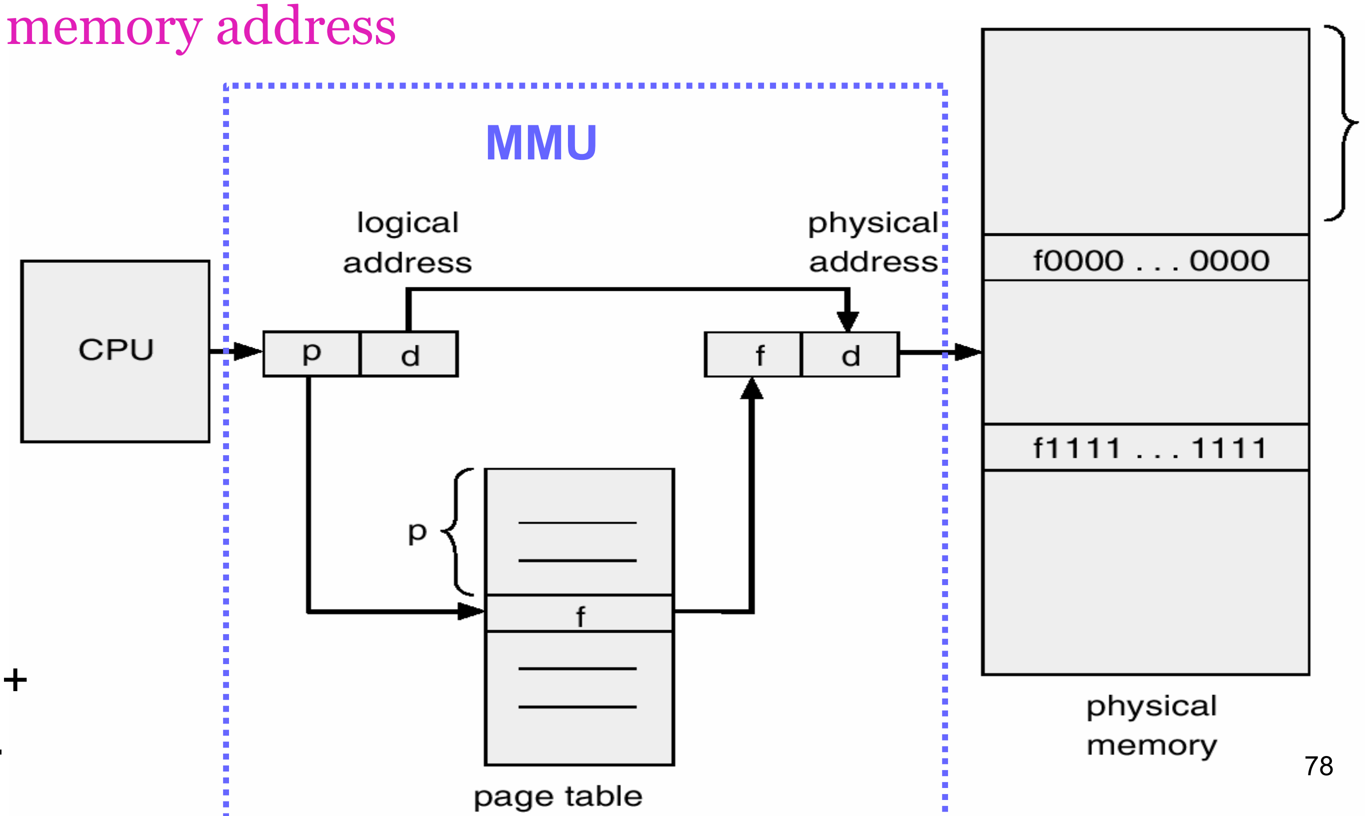


Figure 7.9 Assignment of Process Pages to Free Frames

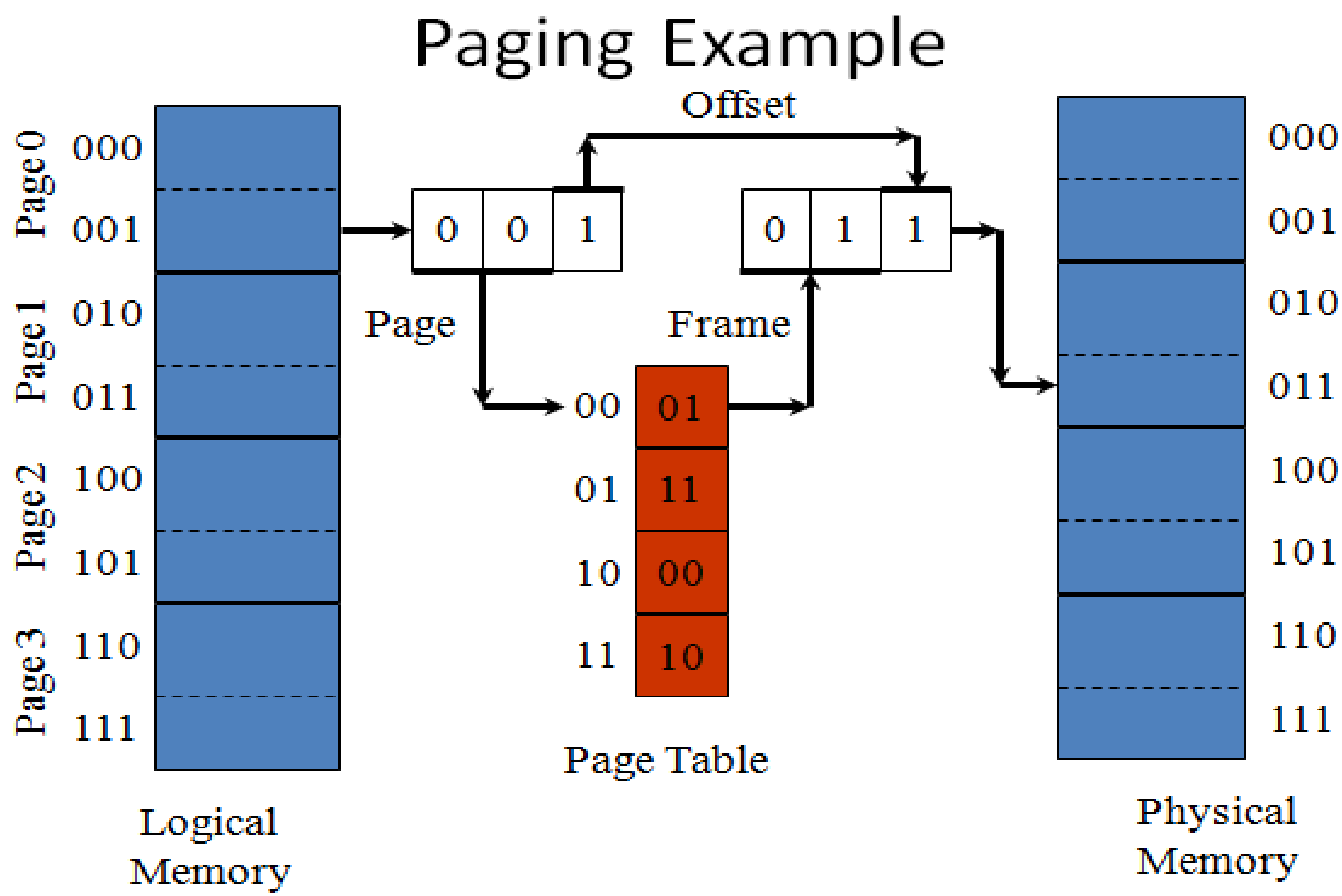
Paging cont…

- The virtual address space of a process is divided into two parts:
  - Page number (p): used as an index into a page table which contains base address of each page in physical memory
  - Page offset (d): combined with base address to define the physical memory address



physical address=frame number + frame offset.

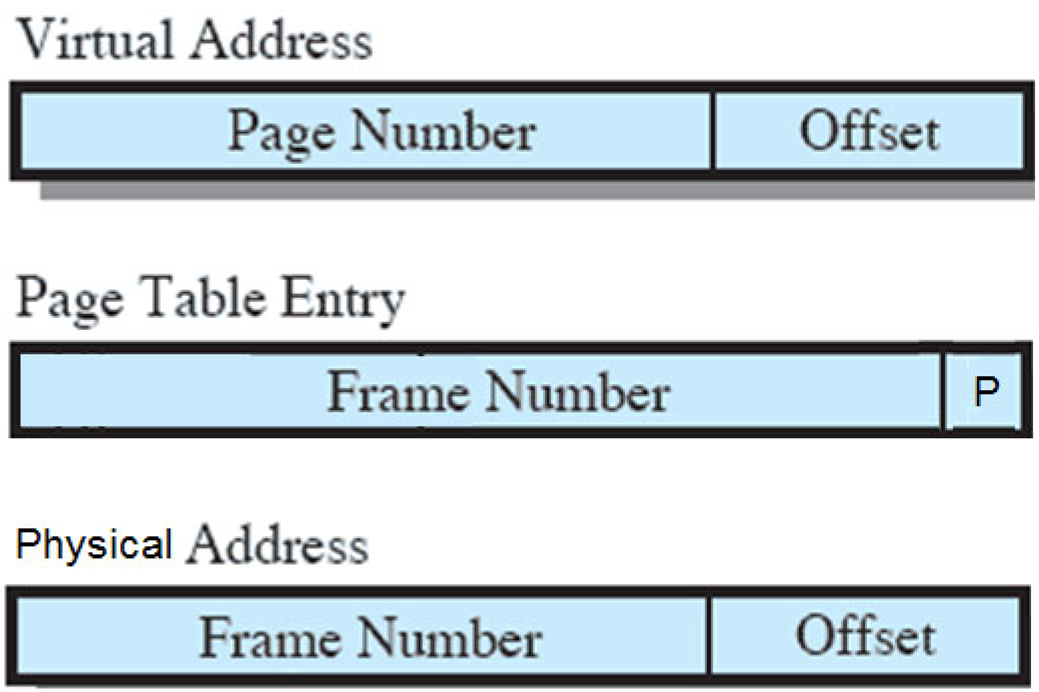
Example:



Paging Example
Offset
Page 0
Page 1
Page 2
Page 3
000
001
010
011
100
101
110
111
0
0
1
0
1
1
Page
Frame
00
01
10
11
01
11
00
10
Page Table
Logical Memory
Physical Memory

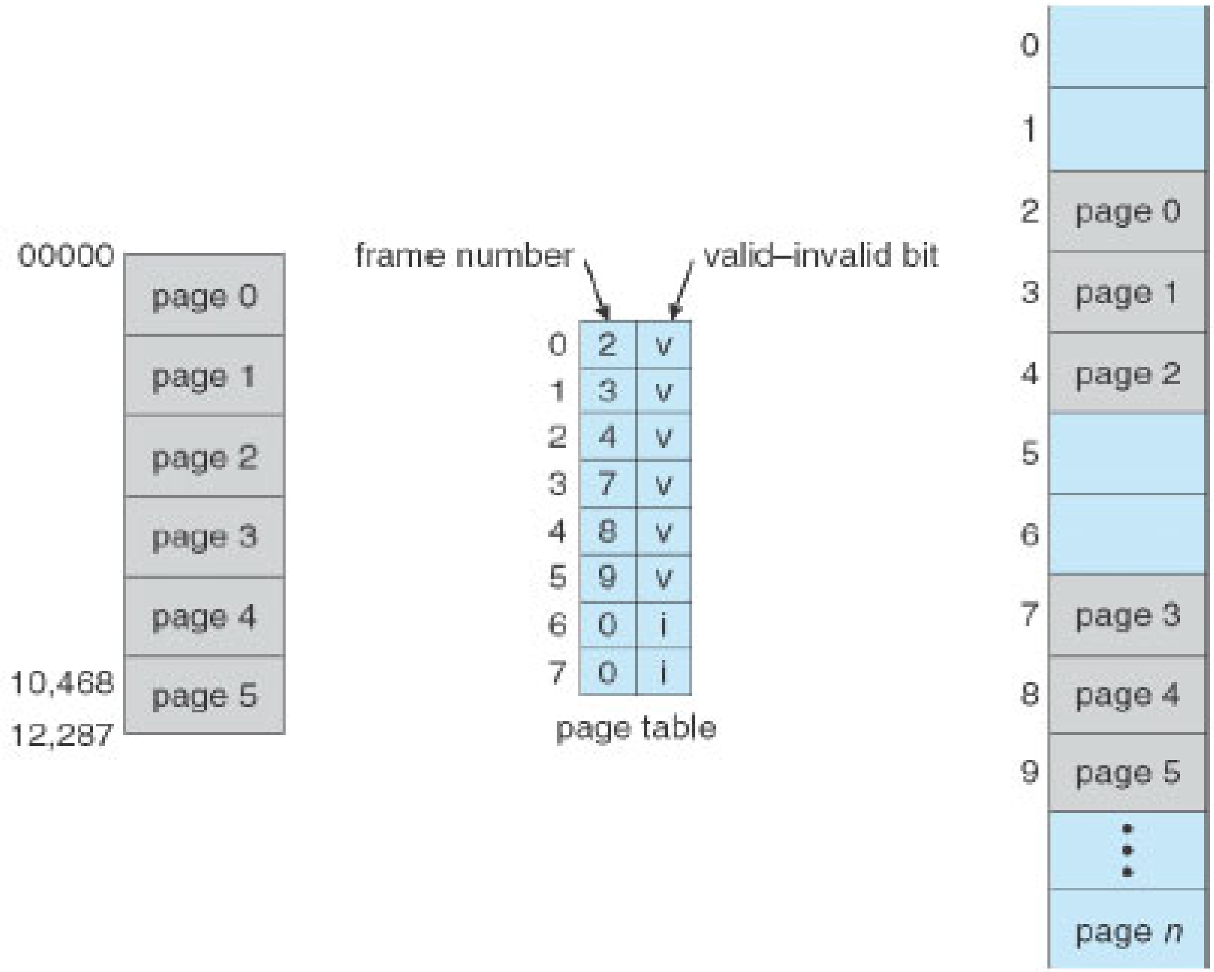
Paging cont…

- OS maintains a page table in memory for each process which contains the frame location for each page in the process.
- The page table is used by processor to produce a physical address.
- The page table may also contain some pages that are not in main memory. Thus, the internal structure of the page table consists of the frame number of the corresponding page in main memory and one extra bits to indicate page availability:
  - P(resent): whether the page is in main memory (v: p=1) or not (i: p=0)



Paging cont…

- Initially valid/invalid bit is set to 0 on all entries. During address translation, if valid/invalid bit in page table entry is 0 ⇒ page fault.



Example:

P1 PT

Page	Frame
0	5
1	12
2	15
3	7
4	22

P2 PT

Page	Frame
0	10
1	18
2	1
3	11

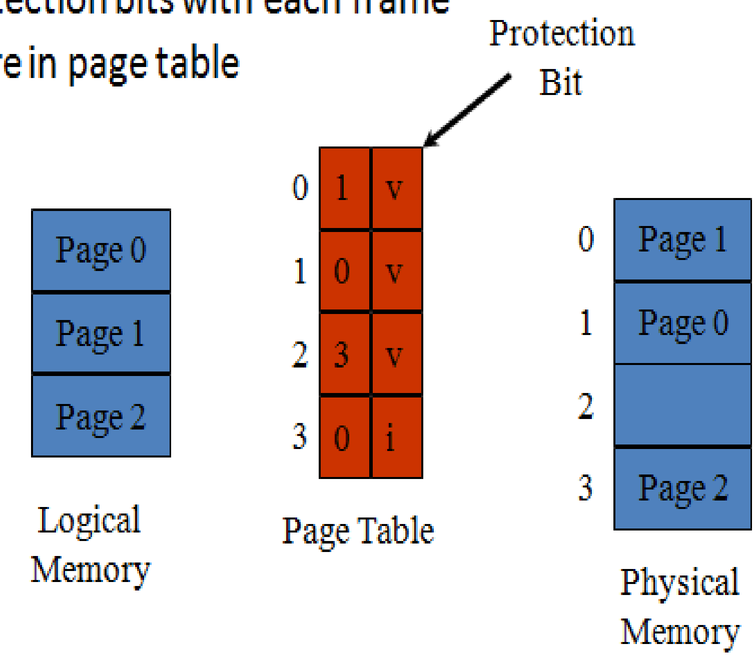
Memory

Frame	Contents
0	
1	P2/Page2
2	
3	
4	
5	P1/Page0
6	
7	P1/Page3
8	
9	
10	P2/Page0
11	P2/Page3
12	P1/Page1
13	
14	
15	P1/Page2

Protection

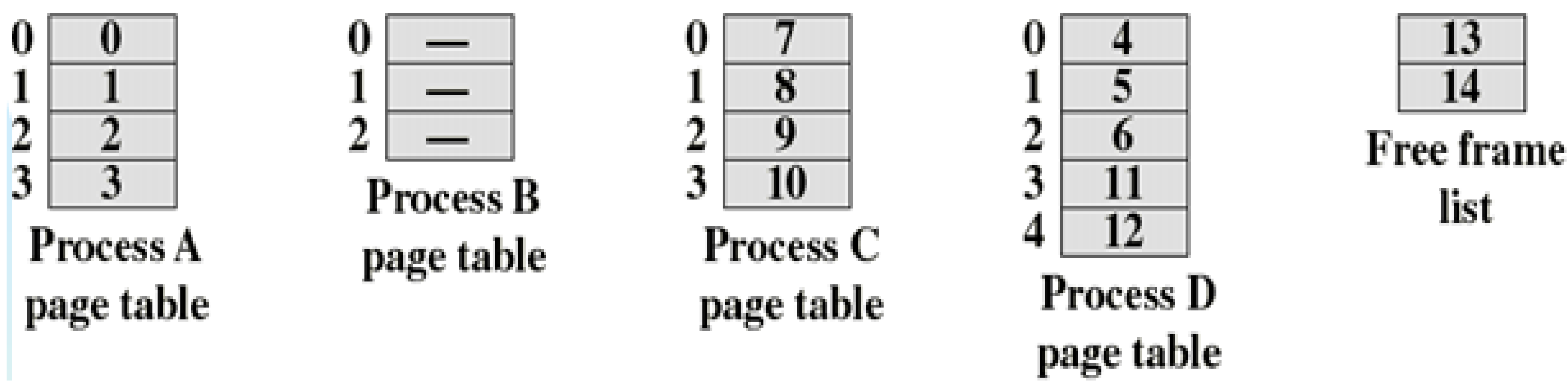
Protection bits with each frame

Store in page table



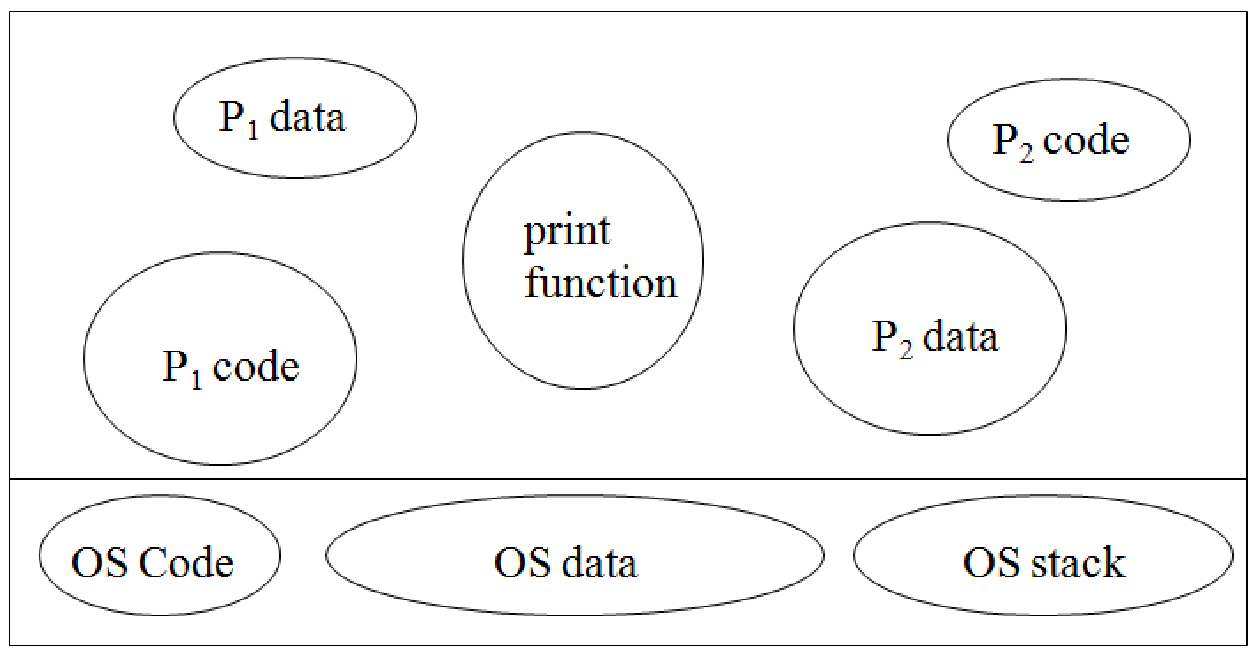
Paging cont…

- The mapping of virtual-to-physical addresses is done on every single memory reference, which can slow things down considerably. A process can no longer directly access memory, but has to go through the MMU for address mapping.
- Each process has its own virtual address space, hence has its own page table.



Segmentation

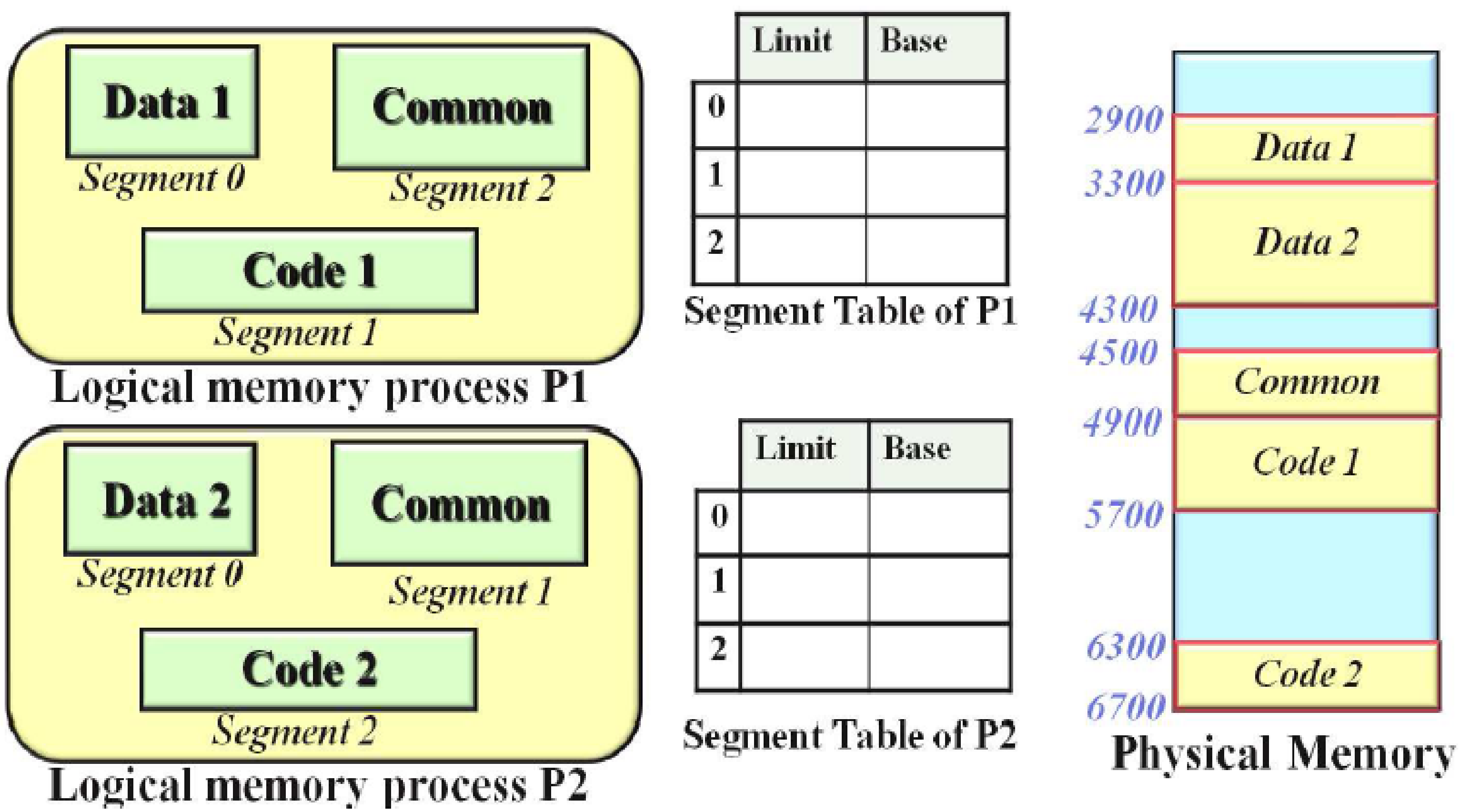
- Segmentation breaks program into logical pieces called segments. Each segment is a grouping of related information:
  - Data segments for each process
  - Code segments for each process
  - Data segments for the OS
  - etc.



logical address space

Segmentation cont…

- Segmentation is very similar to paging **but** the segments can be variable in size.
- Like page tables, each process usually gets its own segment table.



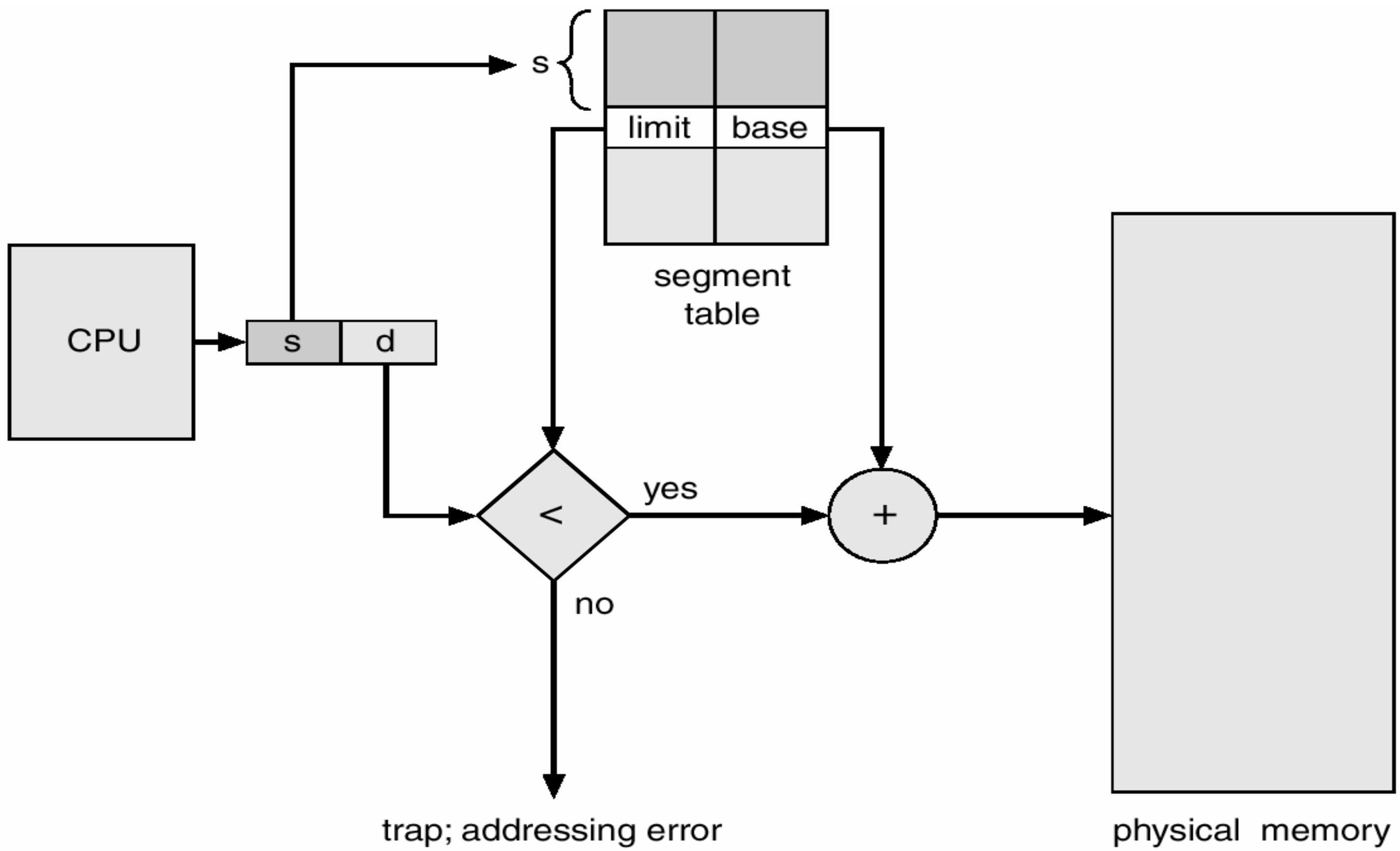
Segmentation cont…

- Use segment number to index into segment table. Each table entry contains:
  - Base address: the physical address of the start of the segment in memory
  - Limit address: the size/length of the segment

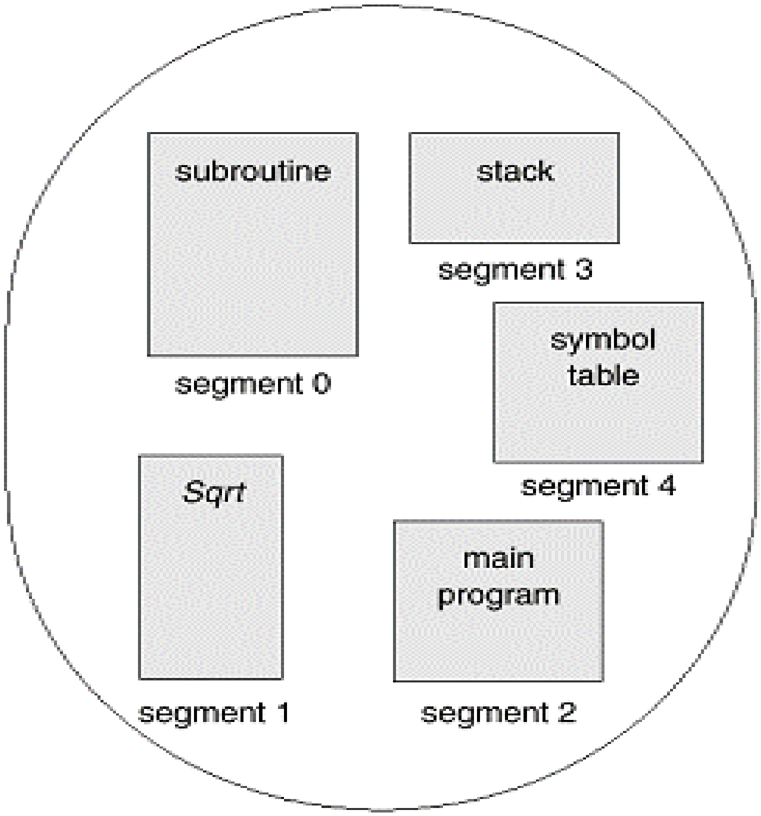
- MMU maintains these two hardware registers for each segment. These registers are used to protect segments from interfering with one another.

Segmentation addressing

- Checks the offset against a limit.
- Add the offset to the base and generate the physical address.



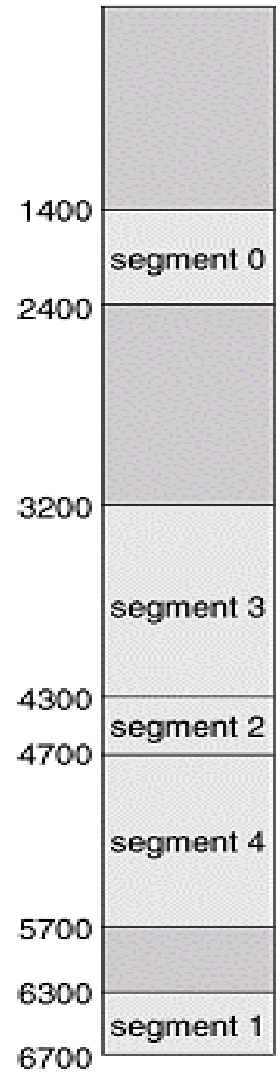
Example:



logical address space

	limit	base
0	1000	1400
1	400	6300
2	400	4300
3	1100	3200
4	1000	4700

segment table



physical memory

Segmentation Issues

- Entire segment is either in memory or on disk
- Variable sized segments leads to external fragmentation in memory
- Must find a space big enough to place segment into
- May need to swap out some segments to bring a new segment in