Memory Management

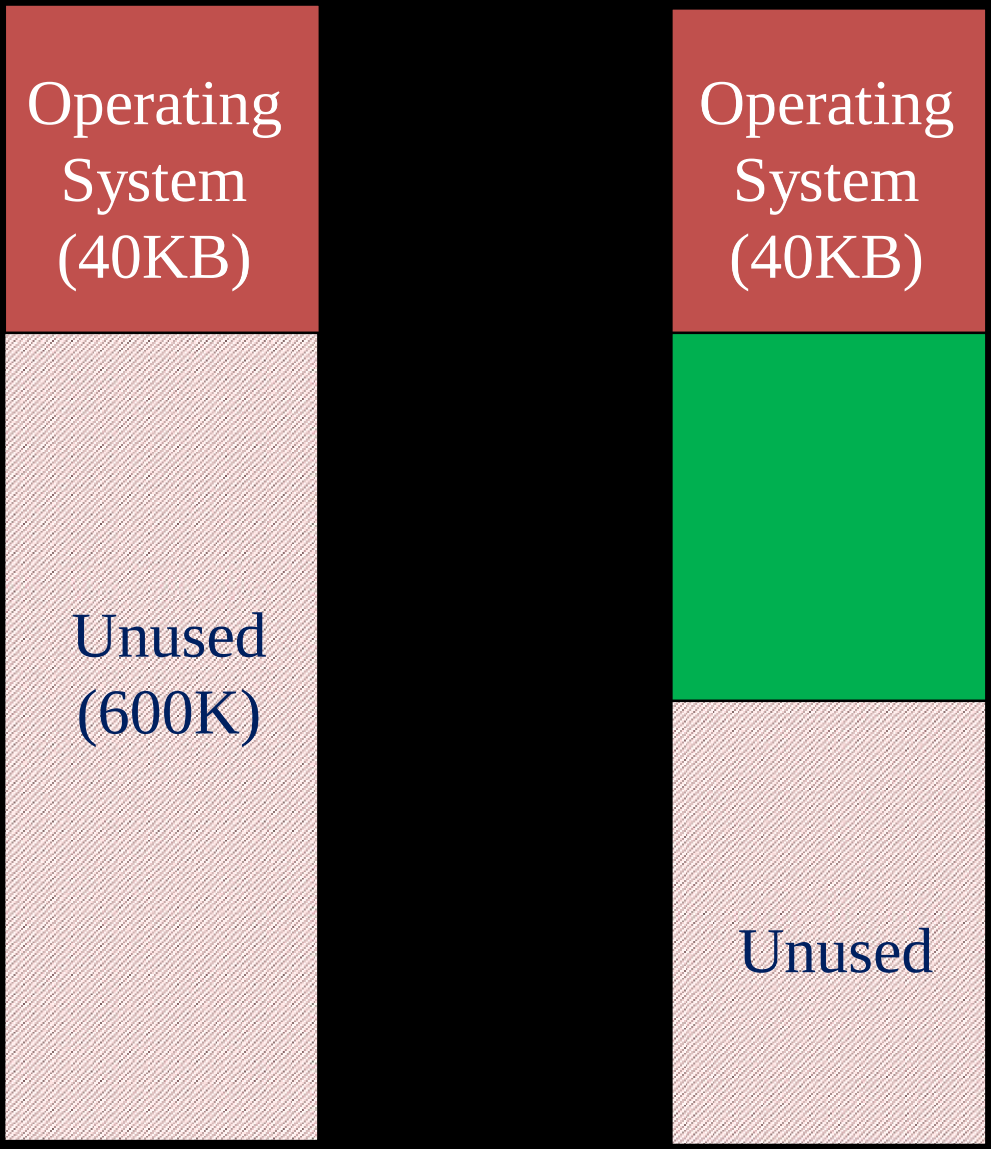
Memory Management - Hierarchy

- Registers
- Cache
- Main memory (RAM, ROM)
- Secondary storage (hard disk, floppy disk, magnetic tape)

Some Preliminaries:

- Program must be brought (from disk) into memory and placed within a process for it to be run
- Main memory and registers are only storage CPU can access directly
- Register access in one CPU clock (or less)
- Main memory can take many cycles
- Cache sits between main memory and CPU registers
- Protection of memory required to ensure correct operation

Memory Management



Single contiguous allocation

In single contiguous allocation, the whole memory, except some space reserved for OS, is allocated to one process.

Memory Management – Single Continuous Allocation

Advantages

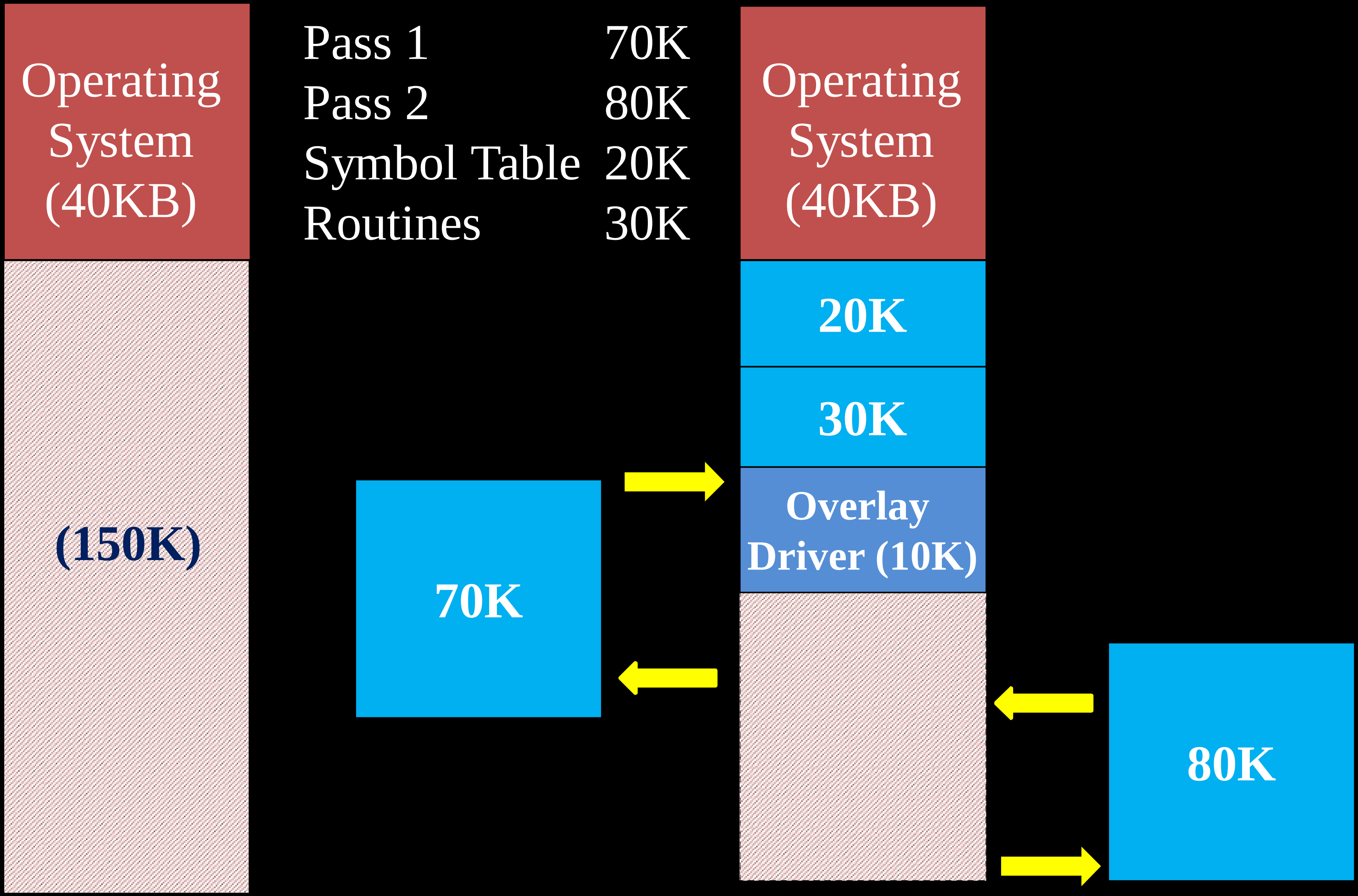
- Simple
- Operating system is small
- No special hardware or software required

Disadvantages

- Wastage of memory
- No multiprogramming
- When I/O device is busy, the memory and CPU is not utilized
- If the address space is more than the available, jobs cannot be run

Memory Management - Overlay

Pass 1	70K
Pass 2	80K
Symbol Table	20K
Routines	30K



Memory Management – Overlay

Keep in memory only those instructions and data that are needed at any given time.

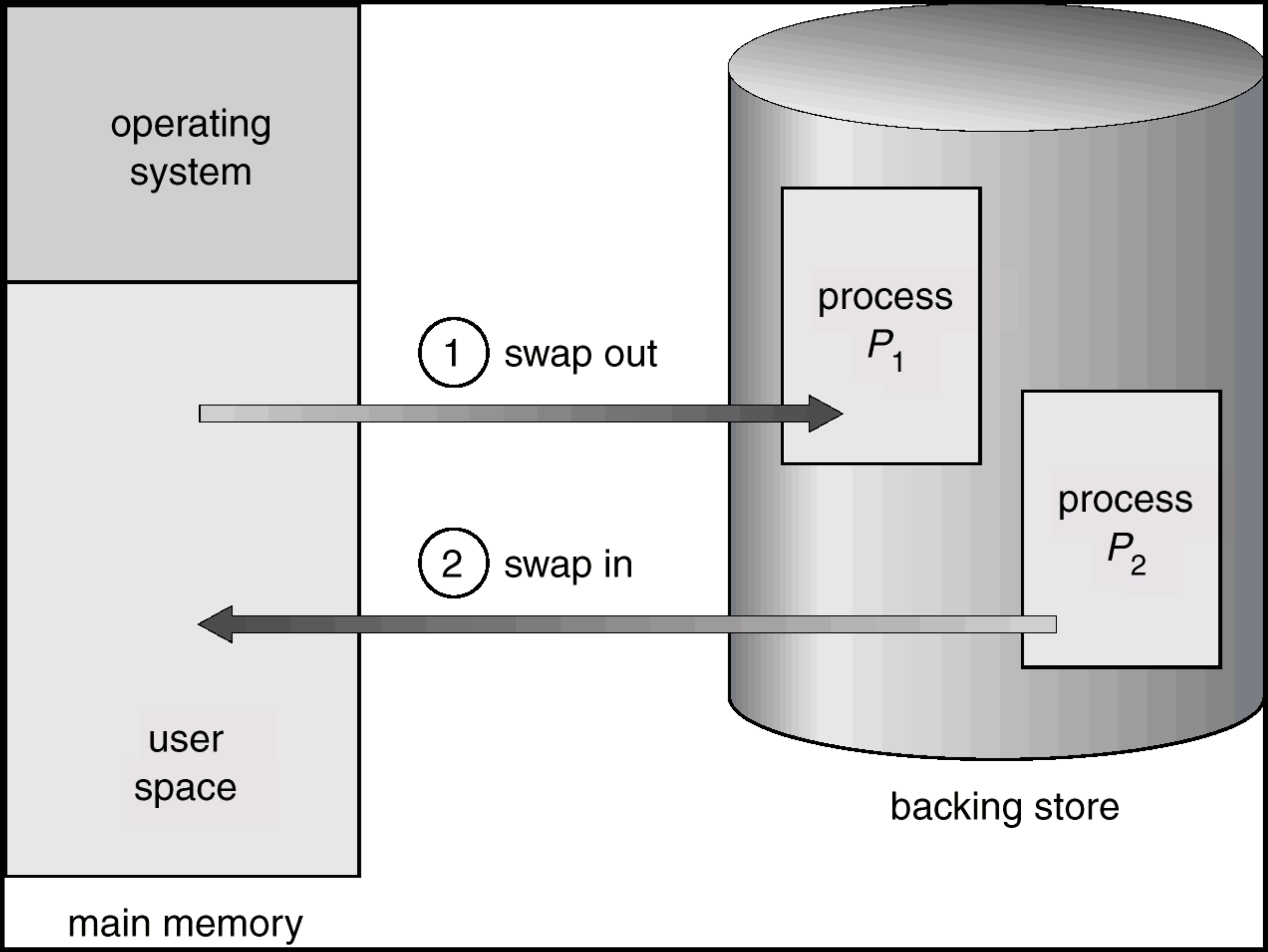
Needed when process is larger than the memory allocated to it.

Implemented by user, no special support needed from operating system, programming design of overlay structure (driver) is complex

Memory Management - Swapping

- A process can be swapped temporarily out of memory to a backing store, and then brought back into memory for continued execution.
- Backing store – fast disk large enough to accommodate 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; total transfer time is directly proportional to the amount of memory swapped.
- Modified versions of swapping are found on many systems, i.e., UNIX, Linux, and Windows.

Memory Management - Swapping



Memory Management - Swapping

A process of size 200 MB needs to be swapped in but there is no space in memory. Another process of size 250 MB is lying idle in memory and therefore, it can be swapped out. If average latency time of HDD is 10 ms and transfer rate is 60 MB / s, how much swap time is required?

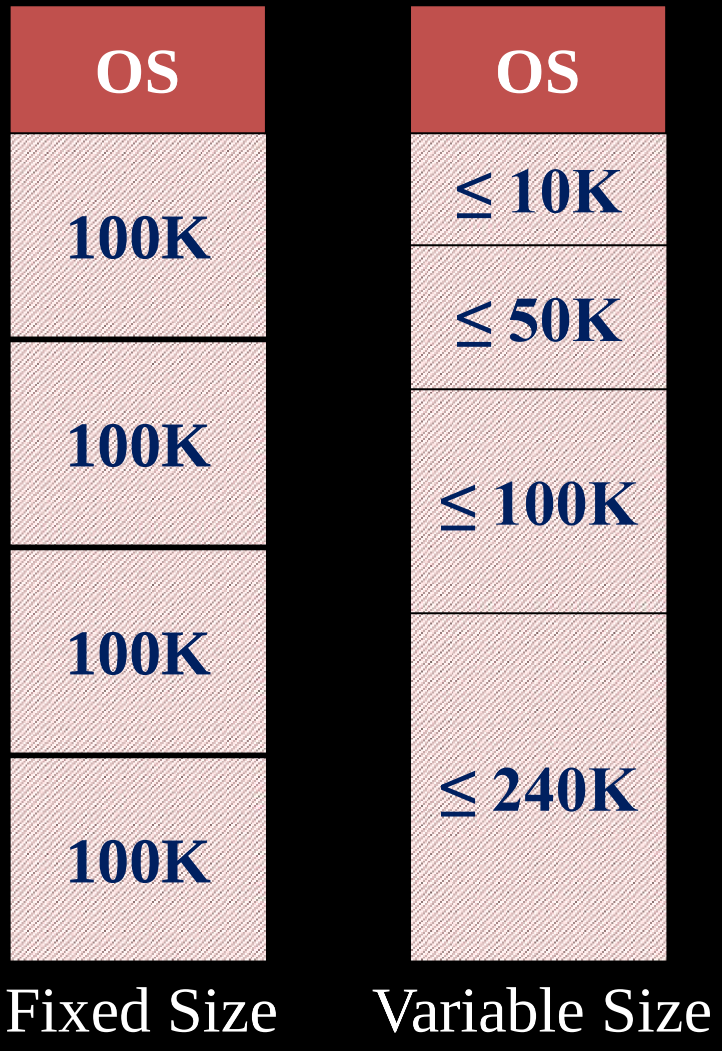
The transfer time of the process to be swapped in = 200 / 60 = 3.34 s = 3340 ms. Therefore, Swap-in time required = 3350 ms. The transfer time of the process to be swapped out = 250 / 60 = 4.17 s = 4170 ms. Therefore, Swap-in time required = 4180 ms. So, total time required for entire swapping process is 3350 + 4180 = 7530 ms.

Memory Management – Partition

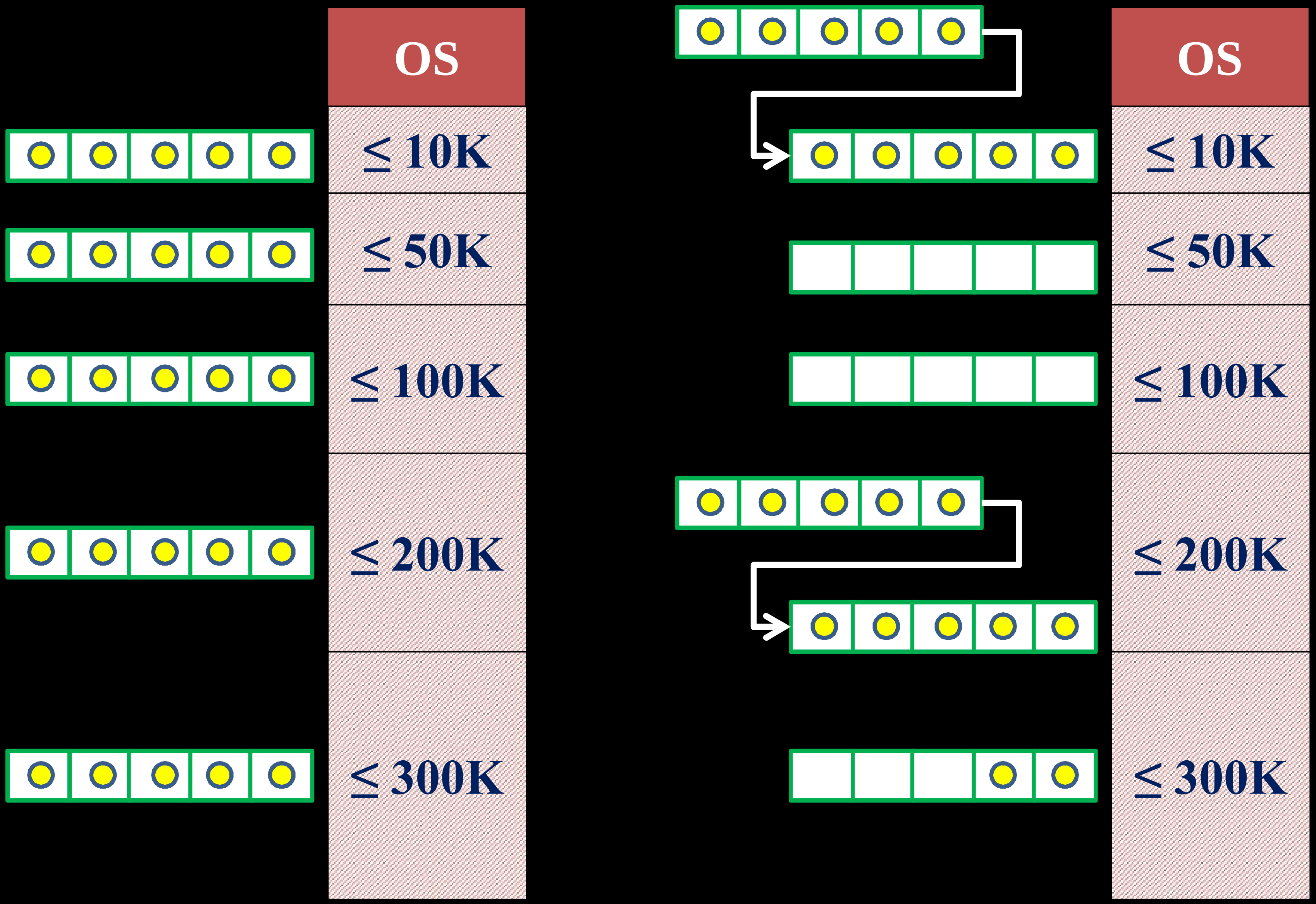
- Simple multiprogramming mechanism
- Memory is divided into partitions
- Two mechanisms
  - Static partitioning
  - Dynamic partitioning

In partitioned allocation, the memory is divided into partitions and a list of all partitions is maintained with all related information about its allocation status

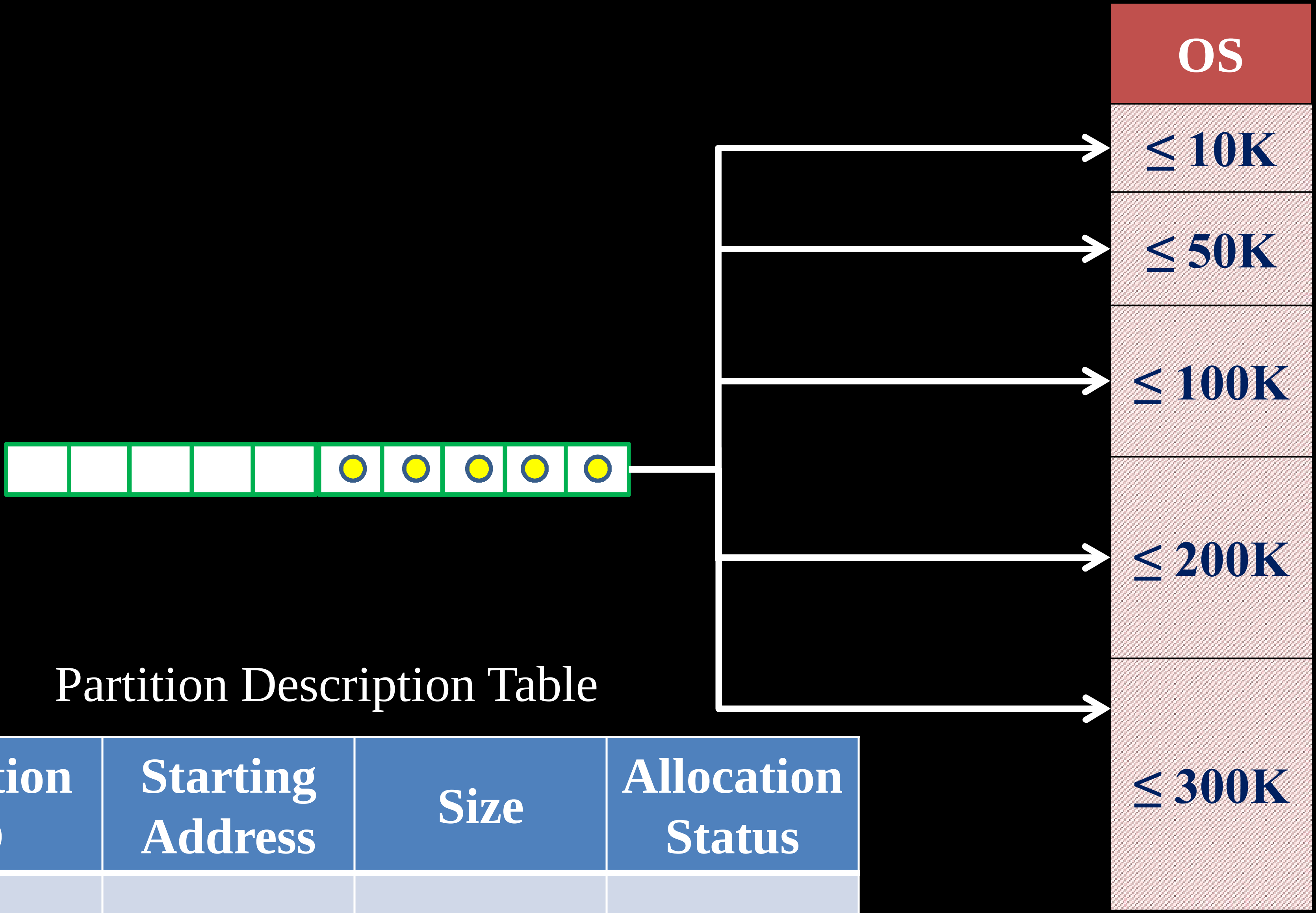
Job size has to be lesser than the partition



Memory Management – Partition



Memory Management – Partition



Partition Description Table

Partition ID	Starting Address	Size	Allocation Status

Memory Management – Address Space

Consider a fixed partitioning scheme with equal-size partitions of 2^{16} bytes and a total main memory size of 2^{24} bytes. A process table is maintained that includes a pointer to a partition for each resident process. How many bits are required for the pointer ?

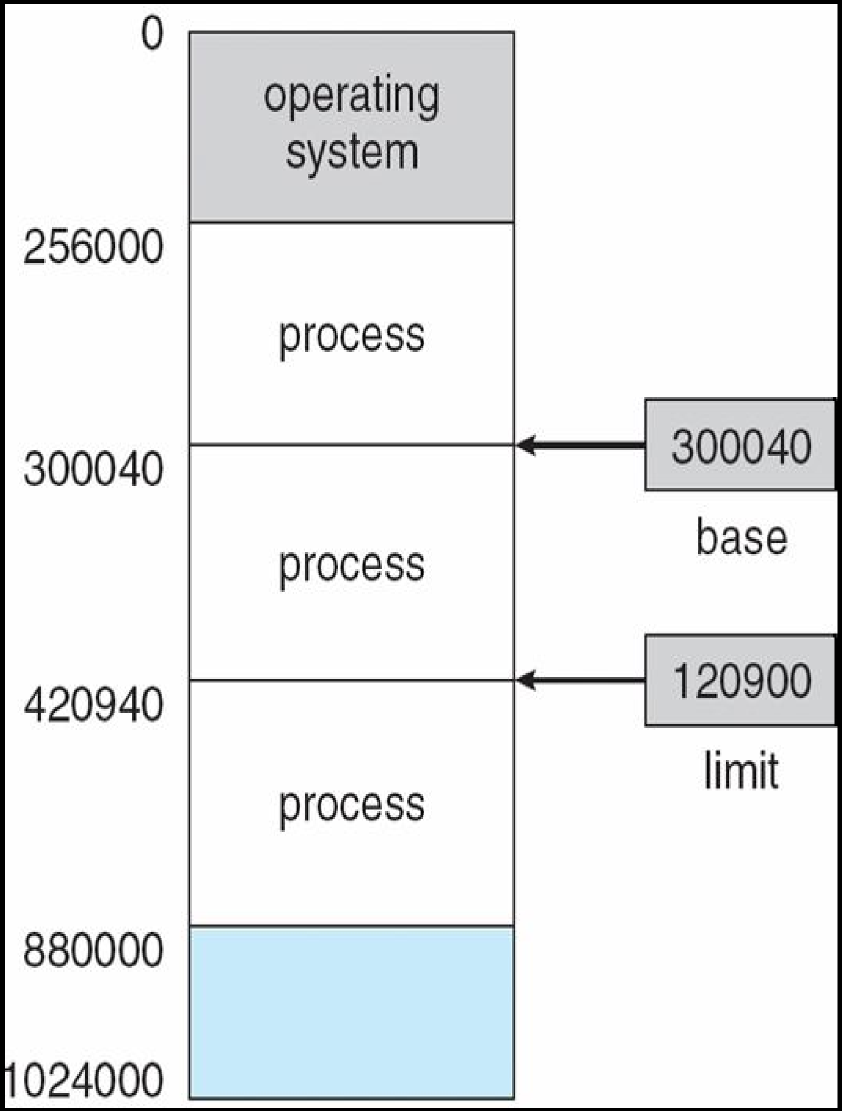
The number of partitions equals the number of bytes of main memory divided by the number of bytes in each partition: 224/216 = 28. Eight bits are needed to identify one of the 28 partitions.

Memory Management – Address Space

The concept of a logical *address space* that is bound to a separate *physical address space* is central to proper Memory management.

- ***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 the same in compile-time and load-time address-binding schemes; logical (virtual) and physical addresses differ in execution time address-binding scheme.



Memory Management – Far Pointer Example

```
#include <stdio.h>
int main() {
     char far *p =(char far *)0x55550005;
     char far *q =(char far *)0x53332225;
      *p = 80;
       (*p)++;
        printf("%d",*q); return 0; }
```

Output of the following program: 81 as both addresses point to same location.
Physical Address = (value of segment register) * 0x10 + (value of offset).
Location pointed to by pointer 'p' is : 0x5555 * 0x10 + 0x0005 = 0x55555
Location pointed to by pointer 'q' is : 0x5333 * 0x10 + 0x2225 = 0x55555
So, p and q both point to the same location 0x55555.

Memory Management – Far Pointer Example

How to normalize far pointer ?

Convert it to its 20-bit address
Use the left 16 bits for segment address and right 4 bits for offset address

What will be normalized address of 500D:9407 ?

Example:

Given a pointer 500D:9407, 20-bit equivalent absolute address is 594D7, which then normalized to 574D:0007

Memory Management – Address Space

What is the difference among logical, relative and physical addresses ?

A **logical address** is a reference to a memory location independent of the current assignment of data to memory; a translation must be made to a physical address before the memory access can be achieved. A **relative address** is a particular example of logical address, in which the address is expressed as a location relative to some known point, usually the beginning of the program. A **physical address,** or absolute address, is an actual location in main memory.

Consider a logical address space of eight pages of 1024 words each, mapped onto a physical memory of 32 frames.

a. How many bits are there in the logical address?
b. How many bits are there in the physical address?

a. Logical address: 13 bits
b. Physical address: 15 bits