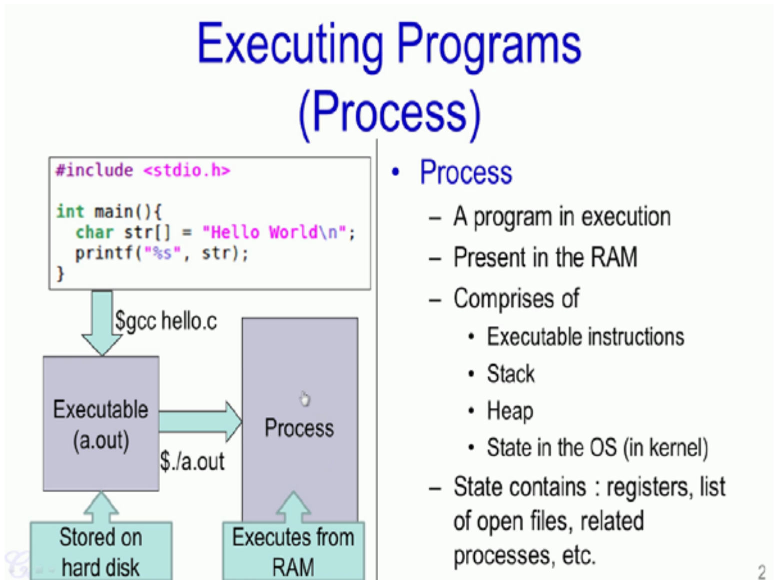
Introduction to Operating Systems
Prof. Chester Rebeiro
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Week - 02
Lecture - 05
Memory Management

Hello. In a previous video we have seen, how the operating system has to manage the CPU because it is probably the most important resource in the system. Arguably, the second most important resource, rather the second most important hardware resource in the system is the memory.

In this video, we will look at how the operating system manages the memory. Essentially, we will see how the operating system manages the RAM present in the system.

(Refer Slide Time: 00:56)



Executing Programs (Process)

```
#include <stdio.h>

int main(){
  char str[] = "Hello World\n";
  printf("%s", str);
}
```

$gcc hello.c
Executable (a.out)
$./a.out
Process
Stored on hard disk
Executes from RAM

- Process
  - A program in execution
  - Present in the RAM
  - Comprises of
    - Executable instructions
    - Stack
    - Heap
    - State in the OS (in kernel)
  - State contains : registers, list of open files, related processes, etc.

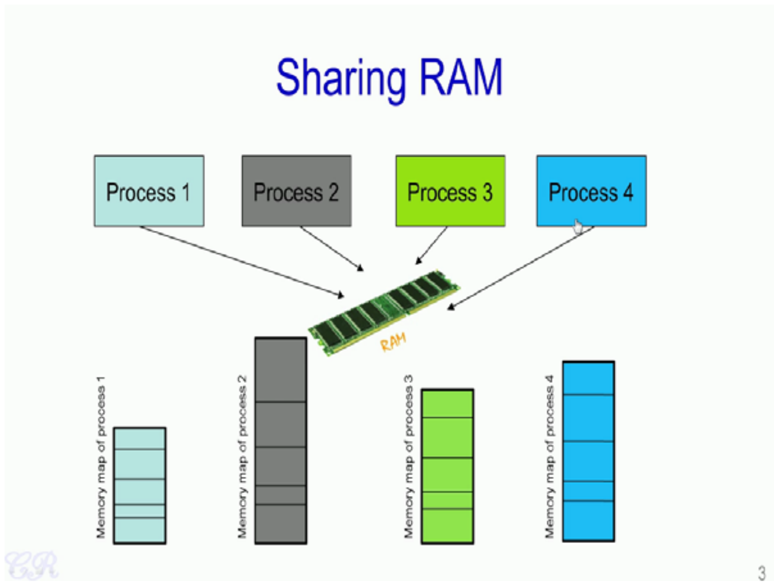
2

Let us review this particular slide again. We have seen that when we take up particular program and this could have been any program written in any language and when we compile it, we will get what is known as an executable.

Now, the executable will be stored on the hard disk and whenever user runs this program or executes this program, it creates what is known as a process. So, this process is created by the operating system and it executes in RAM. So, essentially what the operating system would do is that the executable stored into the hard disk would be loaded into the RAM and then, on execution is passed to this particular program and this program will then execute in the CPU.

So, as we have seen this particular process which is present in memory comprises of several segments, such as the text segment which contains the executable instructions, the stack heap and also, as we have seen some hidden Meta data in the operating system such as registers, list of open files and related processes. So, all these actually constitute the process. Now, what is important for us is this process and it presents in the RAM.

(Refer Slide Time: 02:45)

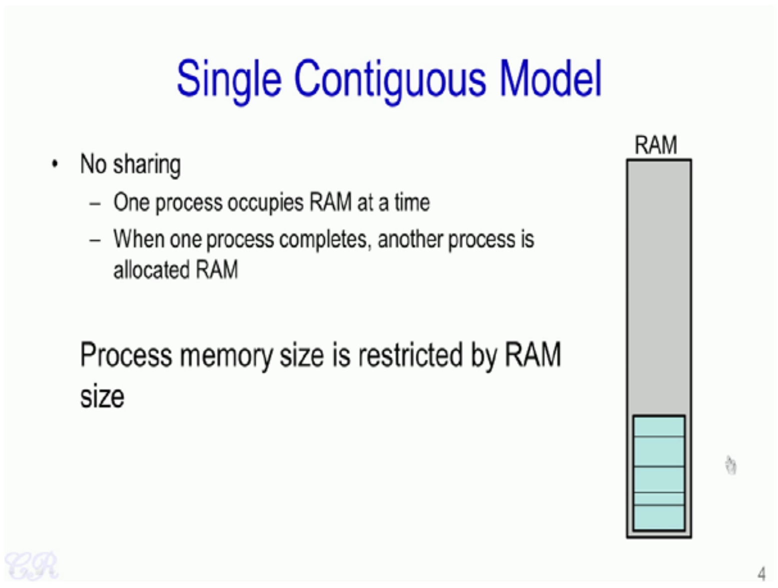


As we know the RAM or Random Access Memory also called as the main memory is a limited resource in the system. So, each system would have something like 4, 8, 16 or 32 GB of RAM. At the same time we may have multiple processes executing almost simultaneously. So, like we have seen in the previous video, these processes may execute in a time sliced manner and if there are multiple processes present in the system, then these processors could also execute in parallel.

However, in all these cases, these processes per and their corresponding memory map should be present in the RAM essentially the memory map corresponding to process 1. The memory map corresponding to process 2, memory map of process 3 and the memory map of process 4 should be present in the RAM, in order that these hope processes execute in parallel or in a multi tasked environment.

Now, what we are going to see is how the operating system manages this resource or in other words, the RAM such that it would allow multiple processes to execute almost simultaneously in the system.

(Refer Slide Time: 04:25)



Single Contiguous Model

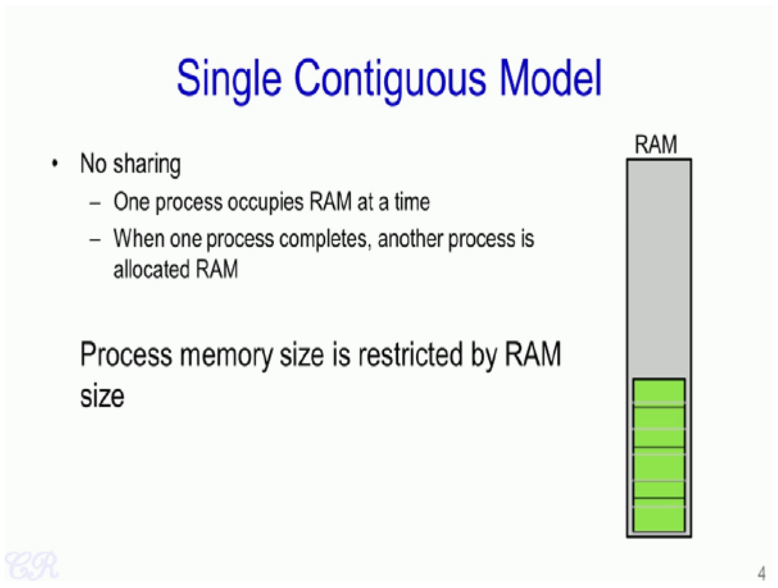
- No sharing
  - One process occupies RAM at a time
  - When one process completes, another process is allocated RAM

Process memory size is restricted by RAM size

4

So, one of the most primitive ways of managing memory especially done for the older on an operating systems is what is known as the single contiguous model. So, essentially in this we have a RAM over here which is the RAM of the system and what is ensured what is done by the operating system is that this RAM is occupied by one process at a time. Essentially at any particular instant there is only one process and it is memory map that is present in the RAM. So, after this process completes executing, the next process will be loaded into the RAM.

(Refer Slide Time: 05:21)



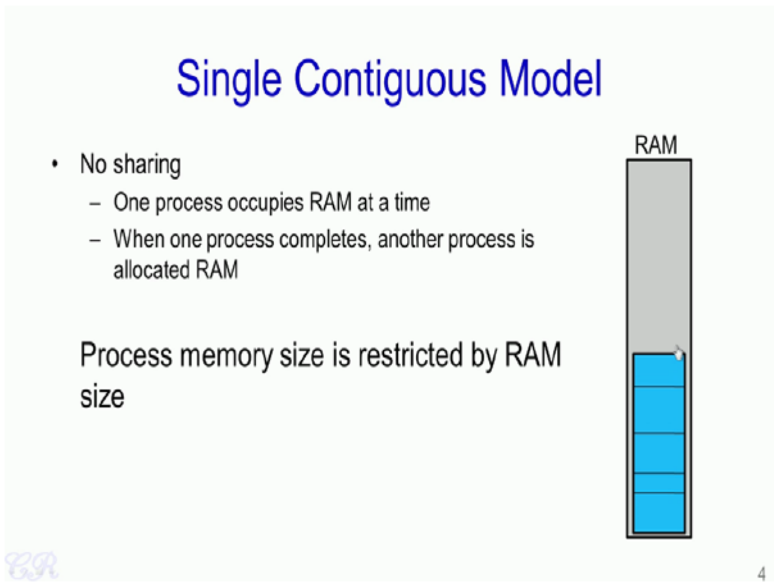
Single Contiguous Model

- No sharing
  - One process occupies RAM at a time
  - When one process completes, another process is allocated RAM

Process memory size is restricted by RAM size

So, the drawbacks are quiet obvious.

(Refer Slide Time: 05:25)



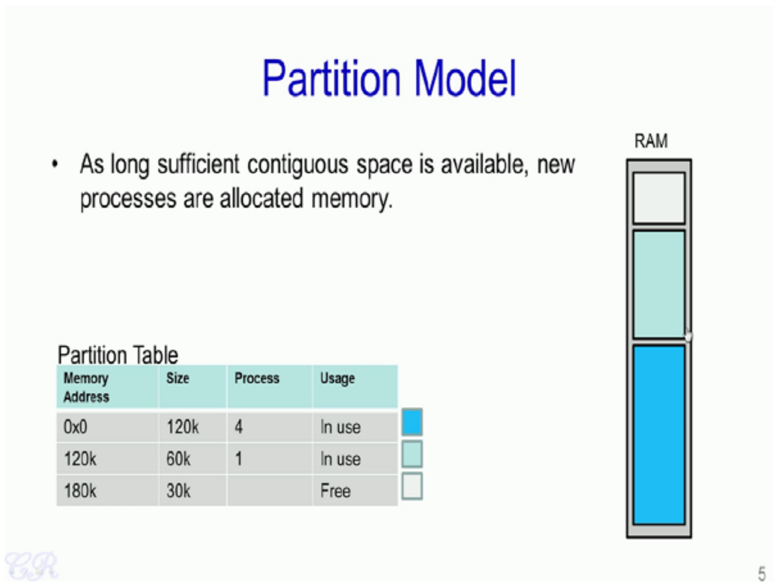
Single Contiguous Model

- No sharing
  - One process occupies RAM at a time
  - When one process completes, another process is allocated RAM

Process memory size is restricted by RAM size

The first says that we are forced to have a very sequential execution. When one process completes, only then the second process could occupy the RAM and so on. Another limitation of this particular model that is a single contiguous model is that the size of the process is limited by the RAM size. For instance, let us say we have RAM which is of say 12 kilo bytes while this seems to be a very small amount of RAM, this size of RAM is quiet common in embedded system. So, given this RAM of say 12 kilo bytes and let us say are process size is of 100 kilo bytes, then it is quite obvious that this process cannot execute using this RAM. Essentially the ram is not sufficient to hold the entire process.

(Refer Slide Time: 06:35)



Partition Model

- As long sufficient contiguous space is available, new processes are allocated memory.

Partition Table

Memory Address	Size	Process	Usage	
0x0	120k	4	In use	🟦
120k	60k	1	In use	🟩
180k	30k		Free	⬜

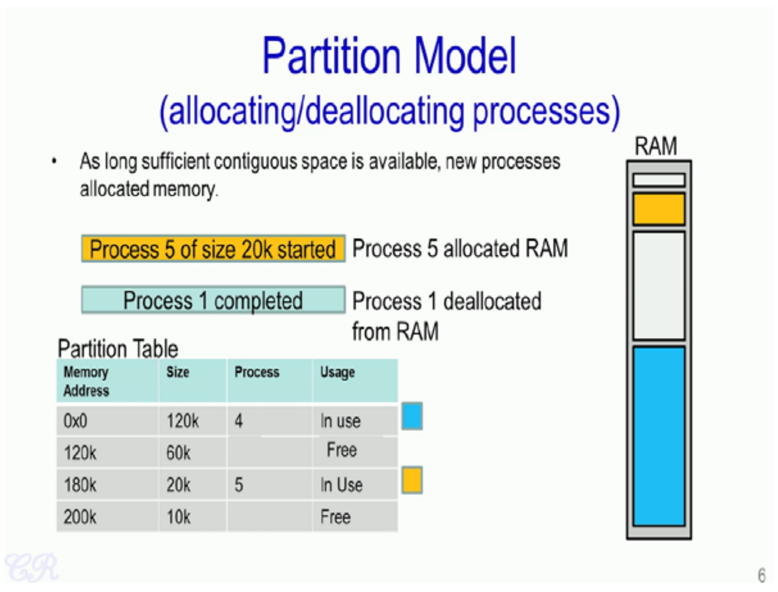
5

In the next model that we will see which is a slight improvement over the single contiguous model is what is known as a partition model. Essentially in this model at any instant of time, we could have multiple processes that occupy the RAM simultaneously. For instance, in this particular case we have two processes. This blue process and the green process that occupy the RAM and therefore, the processor could then execute this process as well as this process either in parallel or in a time sliced manner.

Now, in order to manage such partitions, the operating system would require something known as partition table. So, typically this partition table will also be present in the RAM. It will be present in an area which is not shown over here. So, the partition table would have the base address of process, the size of the process and process identifier. For instance, the blue process has a memory or a base address of 0x0 indicating that is starting from zeroth location in the RAM and this process has a size of 120 kilo bytes. So, 120 kilo bytes means up to this point.

There is also a flag known as the usage flag which mentions whether this particular area in RAM is in use or it is free. For instance, let us take process one that is this one, the green one over here shown over here. This process one starts at the memory location 120K that is this point and has a size of 60K. So, it extends up to 180K and this area is also in use. Now, there is another entry in the partition table which is specified as free. So, this starts at 180K and extends for a size of 30K. So, this corresponds to this white area over here. The operating system could possibly use this free memory to run perhaps the third process and therefore, would be able to have three processes present in the RAM at the same time.

(Refer Slide Time: 09:34)



Partition Model
(allocating/deallocating processes)

- As long sufficient contiguous space is available, new processes allocated memory.

Process 5 of size 20k started — Process 5 allocated RAM

Process 1 completed — Process 1 deallocated from RAM

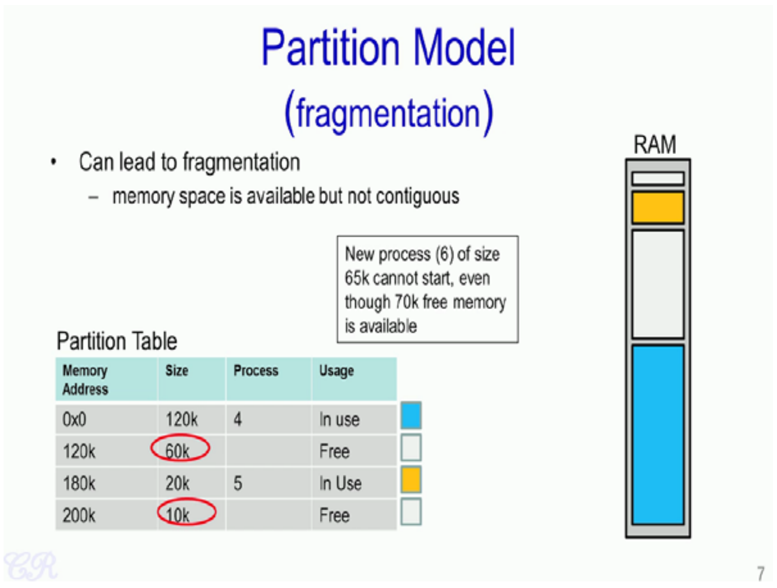
Partition Table

Memory Address	Size	Process	Usage	
0x0	120k	4	In use	🟦
120k	60k		Free	
180k	20k	5	In Use	🟨
200k	10k		Free	

Let us say new process with the identifier 5 has just being created. So, the operating system would look into the partition table and create a particular entry for this new process. Now, since this process requires size of 20 kilo bytes of RAM that the operating system with allocate 20 kilo bytes of RAM for that process. So, this allocation is done from the free space and as we see over here, the third process now gets end present in the RAM while the free space reduces in size. So, we now have the 10 kilo bytes free space.

Similarly, whenever process completes execution for instance when process one completes its execution, the area in RAM that it holds will be de-allocated. Consequently the entry corresponding to the partition table will be free. So, this would lead to free memory of 60K in the RAM and also, this corresponds to this particular area in the RAM which was used by process 1 while this technique of partitioning the RAM and therefore, allowing multiple processes to be allocated in the RAM is quiet easy to do. It could lead what to something known as fragmentation.

(Refer Slide Time: 11:18)



Partition Model (fragmentation)

- Can lead to fragmentation
  - memory space is available but not contiguous

New process (6) of size 65k cannot start, even though 70k free memory is available

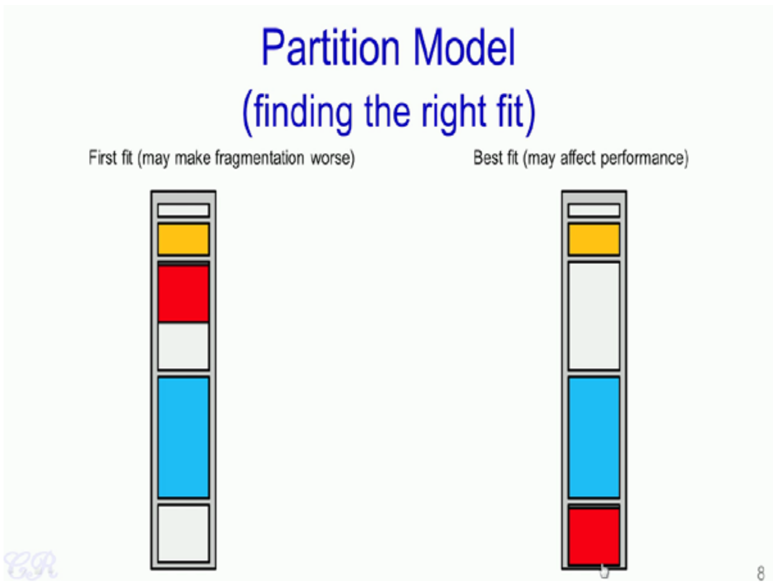
Partition Table

Memory Address	Size	Process	Usage	
0x0	120k	4	In use	🟦
120k	60k		Free	☐
180k	20k	5	In Use	🟨
200k	10k		Free	☐

Let us say this is how our RAM looks and as we seen we have total of 70 kilo bytes of RAM which is free that is we have 60 kilo bytes here plus another 10. So, that is 70 kilo bytes of free RAM memory. However, in spite of having 70 kilo bytes of RAM, new process such as new process with id 6 which has a size 65 k cannot start even though 70 kilo bytes of free RAM is available and the reason for this is that the 70 kilo bytes of free memory is not in contiguous locations.

For instance, we have 60 kilo bytes of free memory present here and 10 kilo bytes of free memory present here. So, individually each of these blocks of free memory is not sufficient to start a new process of 65 kilo bytes. So, this is what is termed as fragmentation and could result in what is known as the underutilization of the RAM.

(Refer Slide Time: 12:37)


Partition Model
(finding the right fit)

First fit (may make fragmentation worse)

Best fit (may affect performance)

8

Now, let us assume that when a new process arrives, there is sufficient amount of free spaces that are present in the RAM memory. So, next the operating system has to decide that which among this free memory the new process should be loaded into. For instance, over here there is a new process which has just arrived and we have 3 blocks of free memory, this one, this one and this one. So, which of these three blocks of free memory in the ram should cross the new process be loaded into?

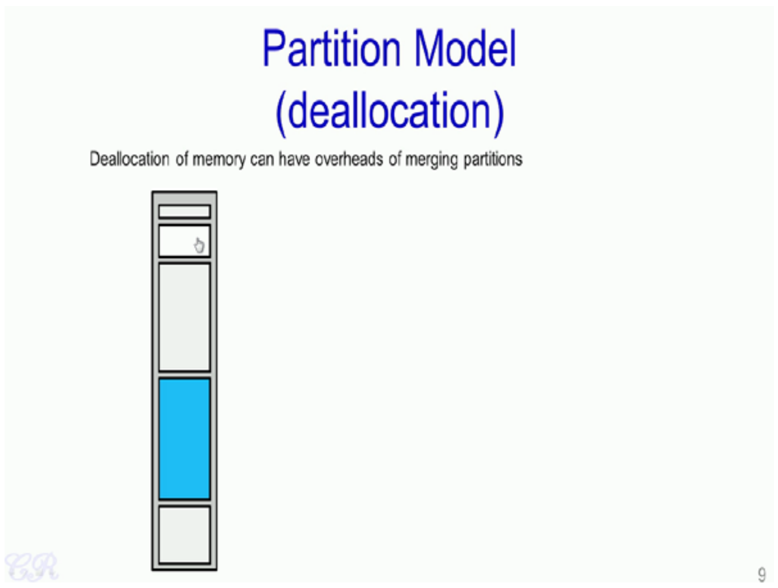
There could be several algorithms to achieve this. So, one of the simplest algorithm is something known as the First Fit. So, with this first fit algorithm what the operating system would do is scan the free blocks in RAM starting from the top and use the first available free block which is at least as large has the process requirement. So, for instance over here this RAM can from the top and see that the first free block is too small for the process to fit in. Therefore, it is not allocated here.

The next free block is large enough to allocate this process. Therefore, the process is allocated here while this first fit allocation algorithm is extremely easy to perform. From an over perspective, it could make fragmentation worse. Essentially with the first fit, what we are doing is that we are breaking the amount of free blocks into smaller and smaller chunks, thus individually each chunk will not be able to data to a single process, thus making the fragmentation worse.

The other algorithm that we will see is known as the best fit algorithm and this performs considerably well with respect to the fragmentation. Essentially it will not fragment the memory as much as the first fit algorithm. So, what happens here is that when a new process enters or is crated, the operating system which scans through all free blocks those are available and choose the block which is the best fit for that process. So, in this particular case as we have seen earlier as well this particular free block is too small to cater to this new process. This free block is too large while the third free block is just correct.

The operating system would allocate this process into this free block as shown over here, while the best fit algorithm efficiently utilizes the available RAM reducing the fragmentation issue, there may be performance hit. Essentially, it may result in deterioration of performance. The reason being that now the operating system has to scan through every free block that is available and needs to make decision about which free block is best for the new process and this would take some time.

(Refer Slide Time: 16:28)



Partition Model (deallocation)

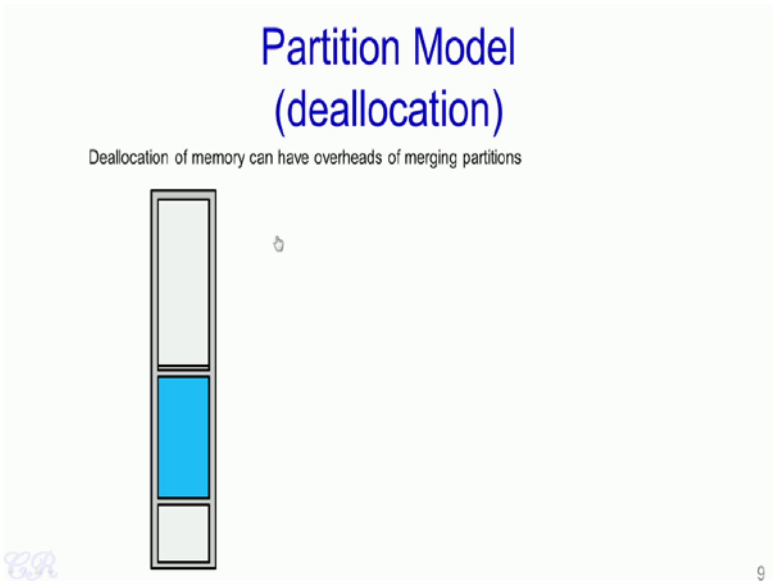
Deallocation of memory can have overheads of merging partitions

9

Another issue with the partitioning model is the case of the deallocation. So, essentially deallocation would occur when a process completes its execution or is terminated, enhance to be removed from the RAM in order to allow new process or another process to execute from the RAM.

So, deallocation would require that new free block to be created and it would result in the free flag set corresponding to this block in the partition table. So, what the operating system now has to do is that it should detect that there are indeed three contiguous free blocks that are present in the RAM and as a result of this, it should detect these three contiguous free blocks and be able to merge these three blocks in to one single block.

(Refer Slide Time: 17:34)

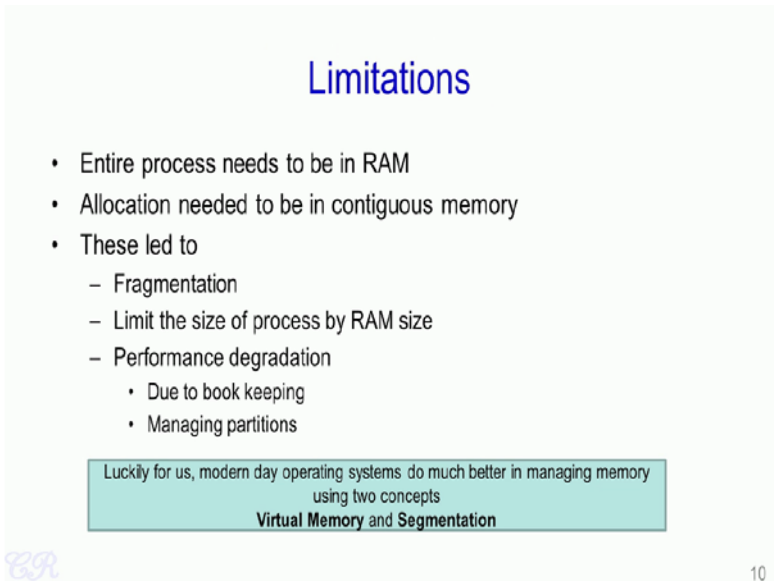


Partition Model (deallocation)

Deallocation of memory can have overheads of merging partitions

The advantage of merging it into one thick free block is that now this larger free block could cater to larger process and thereby reducing the effect of fragmentation.

(Refer Slide Time: 17:51)



Limitations

- Entire process needs to be in RAM
- Allocation needed to be in contiguous memory
- These led to
  - Fragmentation
  - Limit the size of process by RAM size
  - Performance degradation
    - Due to book keeping
    - Managing partitions

Luckily for us, modern day operating systems do much better in managing memory using two concepts
Virtual Memory and **Segmentation**

CR

10

So, thus we have seen that the major limitation of the single contiguous model as well as the partition model is that the entire memory map of the process needs to be present in RAM during its entire execution. So, all allocation for the entire process needed to be in contiguous memory and because of these issues, it had led to fragmentation limit on the size of the process due to base on depending on the RAM size and also performance degradation due to book keeping and also the management of partitions. So, luckily for us modern day operating systems do much better in managing memory. So, most modern day operating systems use two concepts that are virtual memory and segmentation.

In the next few videos, we will look into these memory concepts rather these memory management concepts and we will also see how the Intel X86 processors manages memory.

Thank you.