Memory Management

Main Memory

The main working principle of digital computer is Von-Neumann stored program principle. First of all we have to keep all the information in some storage, mainly known as main memory, and CPU interacts with the main memory only. Therefore, memory management is an important issue while designing a computer system.
On the otherhand, everything cannot be implemented in hardware, otherwise the cost of system will be very high. Therefore some of the tasks are performed by software program. Collection of such software programs are basically known as operating systems. So operating system is viewed as extended machine. Many more functions or instructions are implemented through software routine.

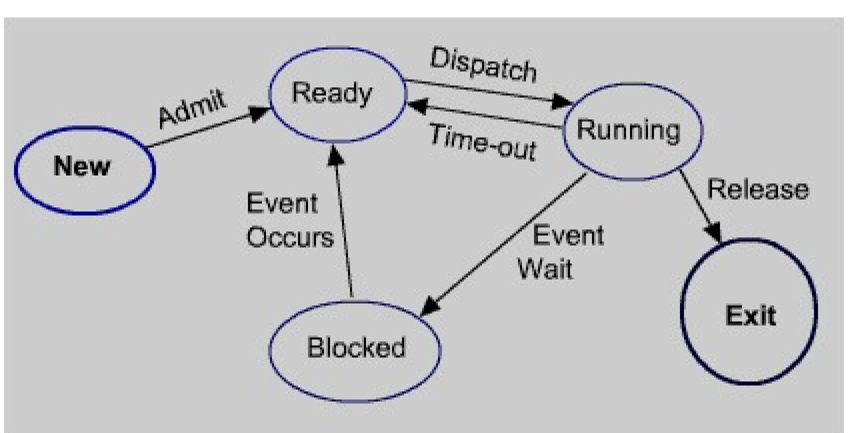
The operating system is mainly memory resistant, i.e., the operating system is loaded into main memory.
Due to that, the main memory of a computer is divided into two parts. One part is reserved for operating system. The other part is for user program. The program currently being executed by the CPU is loaded into the user part of the memory.

In a uni-programming system, the program currently being executed is loaded into the user part of the memory.
In a multiprogramming system, the user part of memory is subdivided to accomodate multiple process. The task of subdivision is carried out dynamically by opearting system and is known as **memory management.**

Efficient memory management is vital in a multiprogramming system. If only a few process are in memory, then for much of the time all of the process will be waiting for I/O and the processor will idle. Thus memory needs to be allocated efficiently to pack as many processes into main memory as possible.
When memory holds multiple processes, then the process can move from one process to another process when one process is waiting. But the processor is so much faster then I/O that it will be common for all the processes in memory to be waiting for I/O. Thus, even with multiprogramming, a processor could be idle most of the time.
Due to the speed mismatch of the processor and I/O device, the status at any point in time is reffered to as a state.

There are five defined state of a process as shown in the figure below.
When a process starts to execute, it is placed in the process queue and it is in the new state. As resources become available, then the process is placed in the ready queue.

At any given time a process may be in one of the following five states.



.

1. New :
A program is admitted to execute, but not yet ready to execute. The operating system will initialize the process by moving it to the ready state.

2.Ready :
The process is ready to execute and is waiting access to the processor.

3.Running :
The process is being executed by the processor. At any given time, only one process is in running state.

4.Waiting : The process is suspended from execution, waiting for some system resource, such as I/O.

5.Exit : The process has terminated and will be destroyed by the operating system.

The processor alternates between executing operating system instructions and executing user processes. While the operating system is in control, it decides which process in the queue sholud be executed next.
A process being executed may be suspended for a variety of reasons. If it is suspended because the process requests I/O, then it is places in the appropriate I/O queue. If it is suspended because of a timeout or because the operating system must attend to processing some of it's task, then it is placed in ready state.
We know that the information of all the process that are in execution must be placed in main memory. Since there is fix amount of memory, so memory management is an important issue.

Memory Management

In an uniprogramming system, main memory is divided into two parts : one part for the operating system and the other part for the program currently being executed.
In multiprogramming system, the user part of memory is subdivided to accomodate multiple processes.

The task of subdivision is carried out dynamically by the operating system and is known as memory management.
In uniprogramming system, only one program is in execution. After complition of one program, another program may start.
In general, most of the programs involve I/O operation. It must take input from some input device and place the result in some output device.

Partition of main memory for uni-program and multi program is shown in figure below.

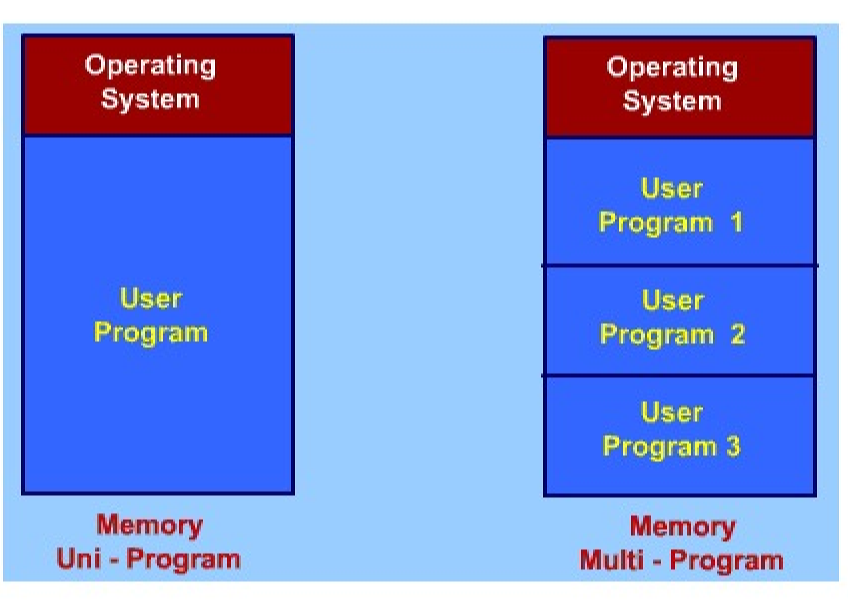


Figure : Partition of Main Memory

To utilize the idle time of CPU, we are shifting the paradigm from uniprogram environment to multiprogram environment.
Since the size of main memory is fixed, it is possible to accomodate only few process in the main memory. If all are waiting for I/O operation, then again CPU remains idle.
To utilize the idle time of CPU, some of the process must be off loaded from the memory and new process must be brought to this memory place. This is known swapping.

What is swapping ?:

The process waiting for some I/O to complete, must stored back in disk.
New ready process is swapped in to main memory as space becomes available.
As process completes, it is moved out of main memory.
If none of the processes in memory are ready,
Swapped out a block process to intermediate queue of blocked process.
Swapped in a ready process from the ready queue.
But swapping is an I/O process, so it also takes time. Instead of remain in idle state of CPU, sometimes it is advantageous to swapped in a ready process and start executing it.
The main question arises where to put a new process in the main memory. It must be done in such a way that the memory is utilized properly.

Partitioning

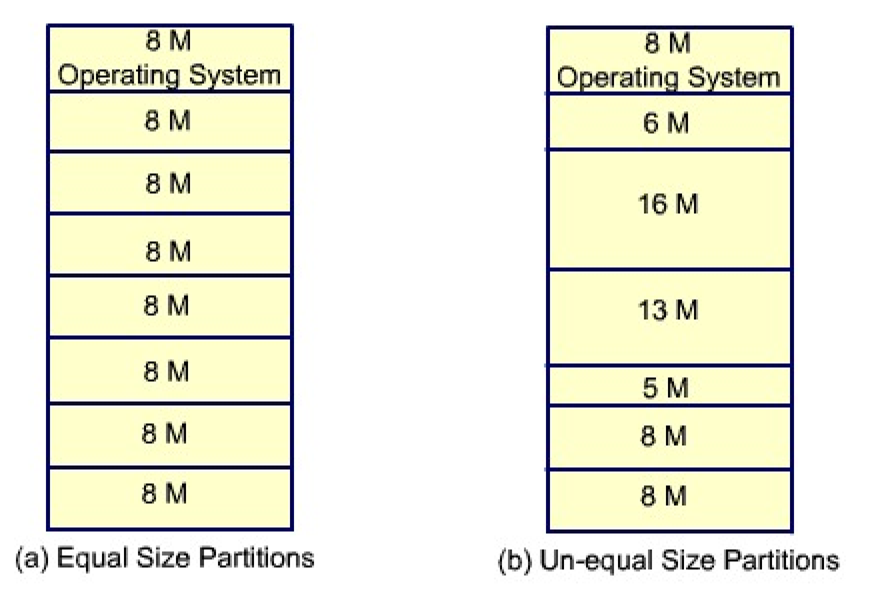
Splitting of memory into sections to allocate processes including operating system.

There are two scheme for partitioning :

1. Fixed size partitions
2. Variable size partitions

Fixed sized pertitions:

The mamory is partitioned to fixed size partition. Although the partitions are of fixed size, they need not be of equal size.
There is a problem of wastage of memory in fixed size even with unequal size. When a process is brought into memory, it is placed in the smallest available partition that will hold it.
Equal size and unequal size partition ofr fixed size partitions of main memory is shown in Figure below.



(a) Equal Size Partitions

(b) Un-equal Size Partitions

Figure : Equal and unequal size partition.

Even with the use of unequal size of partitions, there will be wastage of memory. In most cases, a process will not require exactly as much memory as provided by the partition.

For example, a process that require 5-MB of memory would be placed in the 6-MB partition which is the smallest available partition. In this partition, only 5-MB is used, the remaining 1-MB can not be used by any other process, so it is a wastage. Like this, in every partition we may have some unused memory. The unused portion of memory in each partition is termed as hole.

Variable size Partition:

When a processe is brought into memory, it is allocated exactly as much memory as it requires and no more. In this process it leads to a hole at the end of the memory, which is too small to use. It seems that there will be only one hole at the end, so the waste is less.

But, this is not the only hole that will be present in variable size partition. When all processes are blocked then swap out a process and bring in another process. The new swapped in process may be smaller than the swapped out process. Most likely we will not get two process of same size. So, it will create another whole. If the swap- out and swap-in is occuring more time, then more and more hole will be created, which will lead to more wastage of memory.

There are two simple ways to slightly remove the problem of memory wastage:

Coalesce: Join the adjacent holes into one large hole , so that some process can be accomodated into the hole.

Compaction: From time to time go through memory and move all hole into one free block of memory.
During the execution of process, a process may be swapped in or swapped out many times. it is obvious that a process is not likely to be loaded into the same place in main memory each time it is swapped in. Further more if compaction is used, a process may be shiefted while in main memory.

A process in memory consists of instruction plus data.

The instruction will contain address for memory locations of two types:

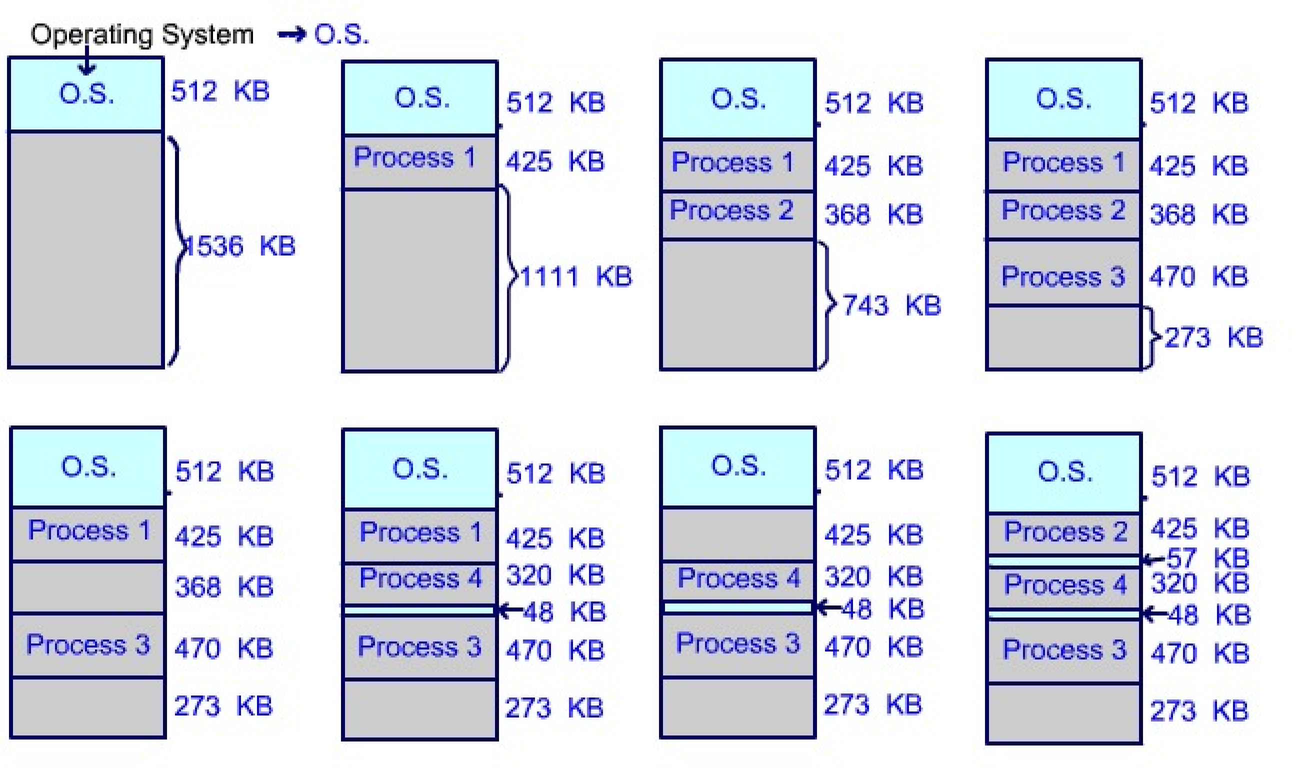
1. **Address of data item**
2. **Address of instructions used for branching instructions**

These addresses will change each time a process is swapped in. To solve this problem, a distinction is made between logical address and physical address.

Logical address is expressed as a location relative to the begining of the program. Instructions in the program contains only logical address.
Physical address is an actual location in main memory.
When the processor executes a process, it automatically converts from logical to physical address by adding the current starting location of the process, called it's base address to each logical address.

Every time the process is swapped in to main memory, the base address may be diffrent depending on the allocation of memory to the process.

Consider a main memory of 2-MB out of which 512-KB is used by the Operating System. Consider three process of size 425-KB, 368-KB and 470-KB and these three process are loaded into the memory. This leaves a hole at the end of the memory. That is too small for a fourth process. At some point none of the process in main memory is ready. The operating system swaps out process-2 which leaves sufficient room for new process of size 320-KB. Since process-4 is smaller then process-2, another hole is created. Later a point is reached at which none of the processes in the main memory is ready, but proces-2, so process-1 is swapped out and process-2 is swapped in there. It will create another hole. In this way it will create lot of small holes in the momory system which will lead to more memory wastage.
The effect of dynamic partitionining that careates more whole during the execution of processes is shown in the Figure below



Figur: The effect of dynamic partitioning