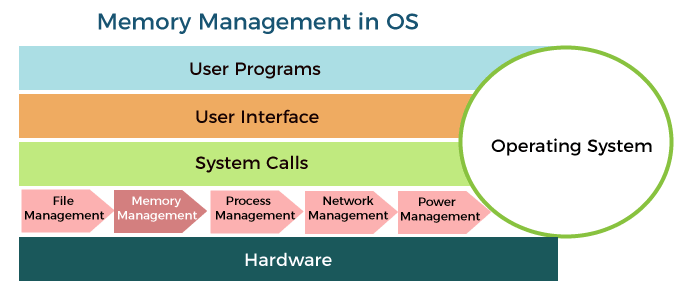
**UNIT-IV**

**Memory Management:**

Background:

The term Memory can be defined as a collection of data in a specific format. It is used to store instructions and processed data. The memory comprises a large array or group of words or bytes, each with its own location. The primary motive of a computer system is to execute programs. These programs, along with the information they access, should be in the main memory during execution. The CPU fetches instructions from memory according to the value of the program counter.

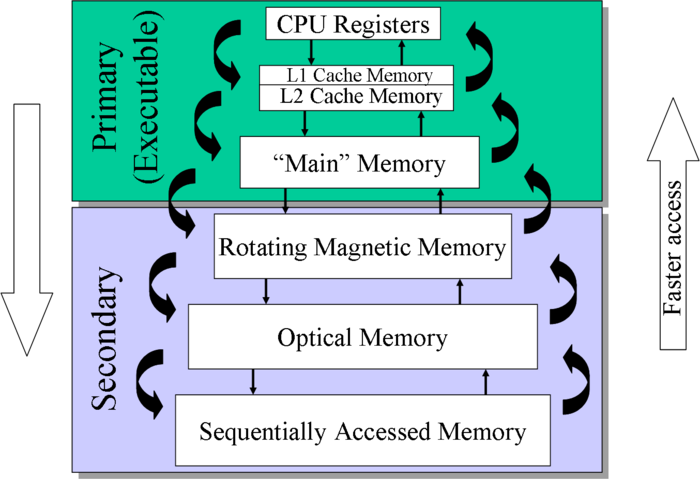
Memory is the important part of the computer that is used to store the data. Its management is critical to the computer system because the amount of main memory available in a computer system is very limited. At any time, many processes are competing for it. Moreover, to increase performance, several processes are executed simultaneously. For this, we must keep several processes in the main memory, so it is even more important to manage them effectively.



**Basic hardware:**

**Memory Hierarchy**

Computers have several different types of memory. This memory is often viewed as a hierarchy as shown below.



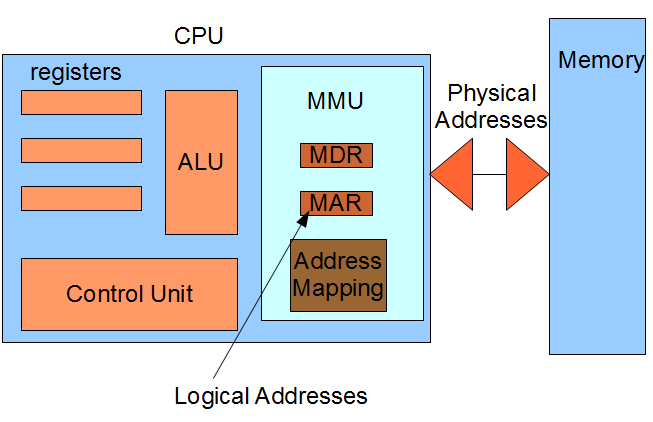
Our main concern here will be the computer’s main or RAM memory. The cache memory is important because it boost’s the speed of accessing memory, but it is managed entirely by the hardware. The rotating magnetic memory or disk memory is used by the [Virtual Memory Management](http://faculty.salina.k-state.edu/tim/ossg/Memory/virt_mem/virt_mem.html#virt-mem).

**Memory Management Unit**

As a program runs, the memory addresses that it uses to reference its data is the [logical address](http://faculty.salina.k-state.edu/tim/ossg/glossary.html#term-logical-address). The real time translation to the [physical address](http://faculty.salina.k-state.edu/tim/ossg/glossary.html#term-physical-address) is performed in hardware by the CPU’s [Memory Management Unit](http://faculty.salina.k-state.edu/tim/ossg/Memory/mem_hw.html#mmu) (MMU). The MMU has two special registers that are accessed by the CPU’s control unit. A data to be sent to main memory or retrieved from memory is stored in the *Memory Data Register* (MDR). The desired logical memory address is stored in the *Memory Address Register* (MAR). The address translation is also called address binding and uses a memory map that is programmed by the operating system.

**Note**

The job of the operating system is to load the appropriate data into the MMU when a processes is started and to respond to the occasional [Page Faults](http://faculty.salina.k-state.edu/tim/ossg/Memory/virt_mem/page_fault.html#page-fault) by loading the needed memory and updating the memory map.



Before memory addresses are loaded on to the system bus, they are translated to physical addresses by the MMU.

**Address Binding:**

The Association of program instruction and data to the actual physical memory locations is called the Address Binding. Let’s consider the following example given below for better understanding.

Consider a program P1 has the set of instruction such that I1, I2, I3, I4, and program counter value is 10, 20, 30, 40 respectively.

Program P1

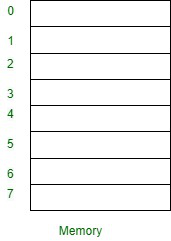
I1 --> 10

I2 --> 20

I3 --> 30

I4 --> 40

Program Counter = 10, 20, 30, 40



**Types of Address Binding :**

Address Binding divided into three types as follows.

1. [Compile-time Address Binding](https://www.geeksforgeeks.org/difference-between-compile-time-and-load-time-address-binding/)
2. [Load time Address Binding](https://www.geeksforgeeks.org/difference-between-compile-time-and-load-time-address-binding/)
3. [Execution time Address Binding](https://www.geeksforgeeks.org/difference-between-compile-time-and-execution-time-address-binding/)

**Compile-time Address Binding :**

* If the compiler is responsible for performing address binding then it is called compile-time address binding.
* It will be done before loading the program into memory.
* The compiler requires interacts with an OS memory manager to perform compile-time address binding.

**Load time Address Binding :**

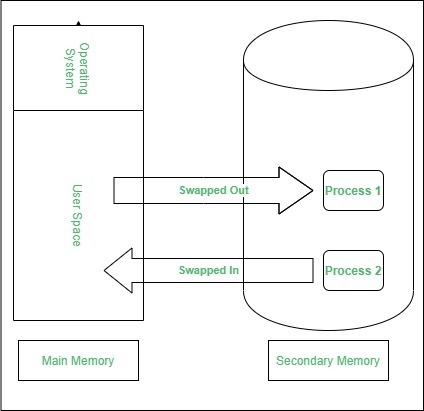
* It will be done after loading the program into memory.
* This type of address binding will be done by the OS memory manager i.e loader.

**Execution time or dynamic Address Binding :**

* It will be postponed even after loading the program into memory.
* The program will be kept on changing the locations in memory until the time of program execution.
* The dynamic type of address binding done by the processor at the time of program execution.

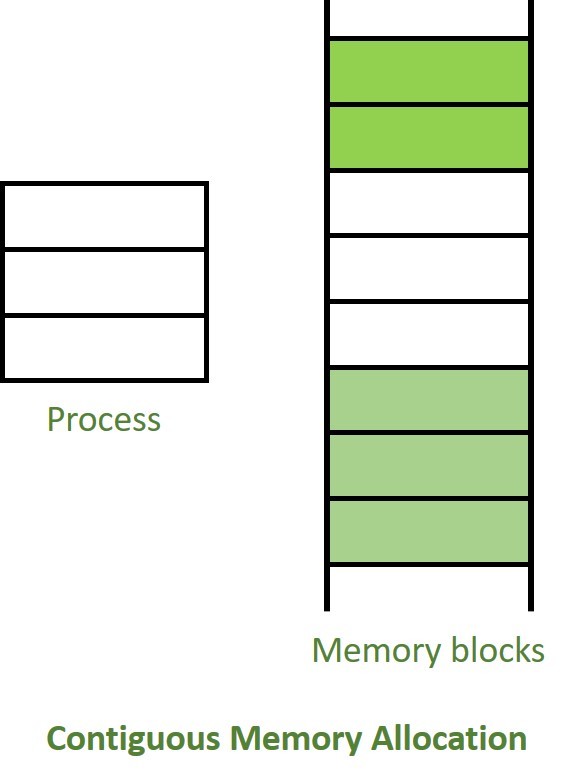
**Swapping:**

When a process is executed it must have resided in memory. Swapping is a process of swap a process temporarily into a secondary memory from the main memory, which is fast as compared to secondary memory. A swapping allows more processes to be run and can be fit into memory at one time. The main part of swapping is transferred time and the total time directly proportional to the amount of memory swapped. Swapping is also known as roll-out, roll in, because if a higher priority process arrives and wants service, the memory manager can swap out the lower priority process and then load and execute the higher priority process. After finishing higher priority work, the lower priority process swapped back in memory and continued to the execution process.



**Contiguous Memory Allocation :**

The main memory should oblige both the operating system and the different client processes.  Therefore, the allocation of memory becomes an important task in the operating system.  The memory is usually divided into two partitions: one for the resident operating system and one for the user processes. We normally need several user processes to reside in memory simultaneously. Therefore, we need to consider how to allocate available memory to the processes that are in the input queue waiting to be brought into memory. In adjacent memory allotment, each process is contained in a single contiguous segment of memory.



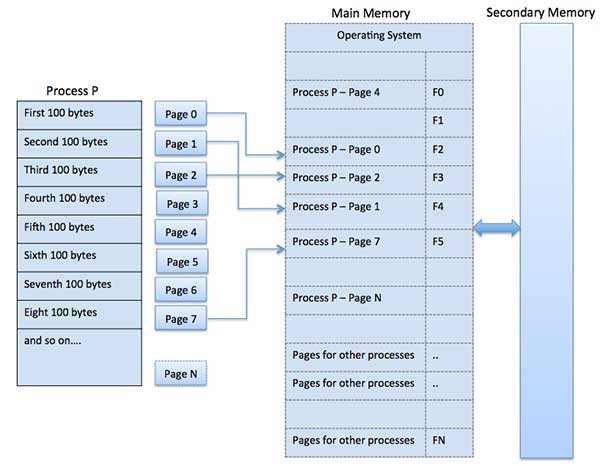
**Paging**

**Basic method:**

A computer can address more memory than the amount physically installed on the system. This extra memory is actually called virtual memory and it is a section of a hard that's set up to emulate the computer's RAM. Paging technique plays an important role in implementing virtual memory.

Paging is a memory management technique in which process address space is broken into blocks of the same size called **pages** (size is power of 2, between 512 bytes and 8192 bytes). The size of the process is measured in the number of pages.

Similarly, main memory is divided into small fixed-sized blocks of (physical) memory called **frames** and the size of a frame is kept the same as that of a page to have optimum utilization of the main memory and to avoid external fragmentation.



## Hardware Support in paging:

Most OS’s allocate a page table for each process. A pointer to the page table is stored with the other register values in the process control block. When the dispatcher is told to start a process, it must reload the user registers and define the correct hardware page table values from the stored user page table.

In the hardware implementation of the page table, it can be implemented as a set of dedicated registers. These registers should be built with very high speed logic to make the paging address translation efficient. Every access to memory must go through the paging map. The use of registers is satisfactory if the page table is small. For contemporary computers, where page table is very large, the use of fast registers to implement the page table is not feasible. Hence, the page table is kept in main memory and a page table base register (PTBR) points to the page table. Changing the page tables requires changing only this one register reducing context switch time.

But the time required for accessing the user location is too large. The solution to this problem is to use a special small fast lookup hardware cache called translation look aside buffer (TLB). The TLB is associative, high-speed memory. Each entry in the TLB consists of two parts a key (or a tag) and a value.

The TLB contains only a few of the page table entries. When a logical address is generated by the CPU its page number is presented to the TLB. If the page number is found, its frame number is immediately available and is used to access memory. If the page number is not in the TLB known as TLB miss, a memory reference to the page table must be made. When the frame number is obtained, it can be used to access memory. Some TLB’s allow entries to be wired down, that is they cannot be removed from the TLB. TLB entries for kernel code are wired down.

Some TLB’s store address space identifiers (ASIDS) in each TLB entry. An ASID uniquely identifies each process and is used to provide address space protection for that process. An ASID allows the TLB to contain entries for several different processes simultaneously. If the TLB does not support separate ASIDs, then every time a new page table is selected, the TLB must be flushed or erased to ensure that the next executing process does not use the wrong translation information.

The percentage of times that a particular page number is found in the TLB is called the **hit ratio**. To find the effective memory access time, each case is weighted by its probability.

**Protection:**

Memory protection in a paged environment is accomplished by protection bits associated with each frame. These bits are kept in the page table. One bit can define a page to be read – write or read – only. Every reference to memory goes through the page table to find the correct frame number. When the physical address is being computed, the protection bits can be checked to verify that no writes are being made to a read- only page. An attempt to write to a read – only page causes a hardware trap to the OS.

One additional bit is generally attached to each entry in the page table – a valid – invalid bit. When this bit is set to valid, the associated page is in the process’s logical address space and is thus a legal page. When the bit is set to invalid, the page is not in the process’s logical address space. OS sets this bit for each page to allow or disallow access to the page.

Some systems provide hardware in the form of a page table length register (PTLR) to indicate the size of the page table. This value is checked against every logical address to verify that the address is in the valid range for the process. Failure of this test causes an error trap to the OS.

**Memory Management in UNIX:**

According to Leon, 2007, UNIX is an operating system (OS) is software that manages hardware and software resources of a computer. UNIX was first developed in the 1960s and has been constant development ever since. UNIX is one of the most widely used operating systems in industry, government and education. It is a stable, multi-user, multi-tasking system for servers, desktops and laptops.

## UNIX Memory Management

Memory is an important resource in computer. Memory management is the process of managing the computer memory which consists of primary memory and secondary memory. The goal for memory management is to keep track of which parts of memory are in use and which parts are not in use, to allocate memory to processes when they need it and de-allocate it when they are done. UNIX memory management scheme includes swapping and demand paging.

## Memory Partitioning

The simplest form of memory management is splitting up the main memory into multiple logical spaces called partition. Each partition is used for separate program. There are 2 types of memory partitioning:-

## Single Partition Allocation

Single partition allocation only separates the main memory into operating system and one user process area. Operating system will not able to have virtual memory using single partition. Using single partition is very ineffective because it only allows one process to run in the memory at one time.

## Multiple Partition Allocation

Most of the operating system nowadays is using multiple partitions because it is more flexible. Multiple partition allocation enabled multiple programs run in the main memory at once. Each partition is used for one process.

There are two different forms of multiple partition allocation, which is fixed partitioning and variable partitioning. Fixed partitioning divides memory up into many fixed partitions which cannot be change. However, variable partitioning is more flexible because the partitions vary dynamically in the later as processes come and go. Variable partitioning (Variable memory) has been used in UNIX.

## UNIX Memory Management Strategies

## Overlays

Program will be place into memory during execution. However, a large program will divide into small pieces and loading the pieces as they needed. Overlays will replace the new pieces with the program which is unused. UNIX is using this technique to run a new program by fork the running process which is also known as fork-exec. The overlays technique is illustrated below.

## Swapping

Swapping consists of bringing in each process in physical memory entirely and running it. When the process is no longer in use, the process will be terminated or is swapped out to disk.

The procedure of swapping is illustrated in figure 1.3 below.

Initially only process A is in memory. Then process B is swapped into memory from disk. After that, process A terminates or swapped out to disk. Then process C is swapped into the free space.

## External Fragmentation Problem

The size of each process is different, therefore when the processes is been swapped in and out, there will be a multiple holes in the memory because UNIX is using variable partitioning.

## Solution

There are two techniques to solve this problem, which are memory compaction and fit in the process using algorithms

Compaction

Memory compaction moves all the processes upward as far as possible, so that all the free memory is placed in one large block. However, it is not a good idea because it requires a lots of CPU time.

Most processes will grow as they run, and the processes data segments can grow, as in many programming languages, the process will grow. If there is a hole is next to the process, it can be allocated and the process is allowed to grow into the hole. Therefore it is good to allocate some extra memory whenever a process is swapped in or out.

Algorithms

There are three different types of algorithm can be used to loads the program wherever the memory space is unused, which is first fit, best fit and worst fit.

Algorithms

Descriptions

First Fit

The memory manager scans along the list and allocates the first space to fit the process. First fit is a fast algorithm because it searches as little as possible.

Best Fit

The memory manager scans the whole list and takes the smallest hole that will fit the process. Best fit is slower than first fit because it must search the whole list every time it is called.

Worst Fit

The memory manager scans the whole list and takes the largest available hole, so that the hole broken will be big enough to be useful.

## Virtual Memory

UNIX operating system allows user to fully utilize the physical memory installed in the system as well as part of the hard disk called swap space which have been designated for use by the kernel while the physical memory is insufficient to handle the tasks.

Virtual memory managers will create a virtual address space in secondary memory (hard disk) and it will determine the part of address space to be loaded into physical memory at any given time. The benefit of virtual memory relies on separation of logical and physical memory.

## Demand Paging

Paging is a memory allocation strategy by transferring a fixed-sized unit of the virtual address space called virtual page whenever the page is needed to execute a program. As the size of frames and pages are the same, any logical page can be placed in any physical frame of memory.

Every processes will be logical divided and allocate in the virtual address space. There is a page table in the virtual memory to allocate and keep tracking of the pages to map into the frames.

UNIX will perform page swapping only when the program needs a certain page. This procedure is called demand paging. The page will be paged into the memory only when it is needed to execute. The whole process will not be paged into the memory, only the pages needed are swapped in.

Demand paging decreases the paging time and physical memory needed because only the needed pages will be paged and the reading time of the unused pages can be avoided.

As the example in the figure, only page 6, 7, 8 and 9 is needed in Process A. Therefore, only pages 6, 7, 8 and 9 will be paged into the memory.

## Page Fault Problem

A page fault occurs when a program try to use a page that is not in the memory, due to demand paging will only paged the pages into the memory when it is needed in the memory.

## Solution for Page Fault

The diagram above illustrated the steps in handling page fault. When page fault occurs during program execution, the kernel will first locate the missing page on the backing store (disk). After located the page, it will find a free memory frame in the physical memory and copy into it. The page table will be reset after that and the instruction will be restart.

## Problem- No Free Frames

When all the frames in the memory is been used, the other problem will occurs. This will cause the pages is unable to paged into the memory.

## Solution- Page Replacement Algorithms

Solution for no free frames problem is to find a memory frame that is idle and free the frame using a page replacement algorithm. There are three common types of page replacement algorithm such as First in First out (FIFO), Optimal and Least Recently Used (LRU).

UNIX is using least recently used algorithm for page replacement. The least recently used algorithm replaces the page that has not been used for the longest time, on the assumption that the page will not be needed again. The page table will record every time the page being referenced, and when page replacement is needed, every page will be checked to find the oldest recorded time.

## Conclusion

Every operating system has different memory management. UNIX also has their exclusive memory management strategies to manage the memory resource optimally. UNIX is using multiple and variable partitioning so that the memory can be stored and use more flexible.

UNIX uses overlays and swapping to replace the unused program. However, it is facing external fragmentation problem and solve by loading the program into memory by using best fit algorithm.

Besides, UNIX also fully utilized the virtual memory (physical memory and swap space) by using demand paging. It allows user to store physical memory in the hard disk because the RAM memory was always insufficient.