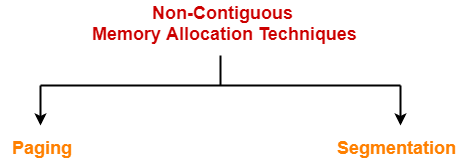
**[Paging | Memory Management | Operating System](https://www.gatevidyalay.com/paging-memory-management-operating-system/)**

**Non-Contiguous Memory Allocation-**

* Non-contiguous memory allocation is a memory allocation technique.
* It allows to store parts of a single process in a non-contiguous fashion.
* Thus, different parts of the same process can be stored at different places in the main memory.

**Techniques-**

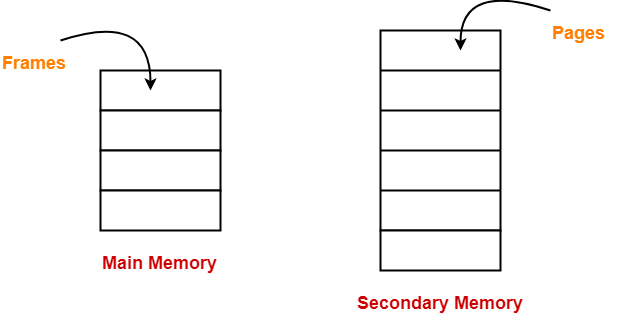
There are two popular techniques used for non-contiguous memory allocation-



1. Paging
2. Segmentation

**Paging-**

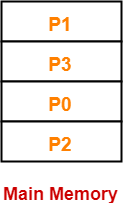
* Paging is a fixed size partitioning scheme.
* In paging, secondary memory and main memory are divided into equal fixed size partitions.
* The partitions of secondary memory are called as **pages**.
* The partitions of main memory are called as **frames**.



* Each process is divided into parts where size of each part is same as page size.
* The size of the last part may be less than the page size.
* The pages of process are stored in the frames of main memory depending upon their availability.

**Example-**

* Consider a process is divided into 4 pages P0, P1, P2 and P3.
* Depending upon the availability, these pages may be stored in the main memory frames in a non-contiguous fashion as shown-



**Translating Logical Address into Physical Address-**

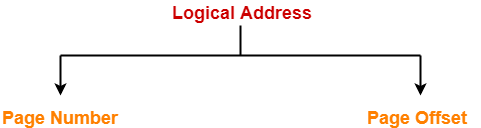
* CPU always generates a logical address.
* A physical address is needed to access the main memory.

Following steps are followed to translate logical address into physical address-

**Step-01:**

CPU generates a logical address consisting of two parts-

1. Page Number
2. Page Offset



* Page Number specifies the specific page of the process from which CPU wants to read the data.
* Page Offset specifies the specific word on the page that CPU wants to read.

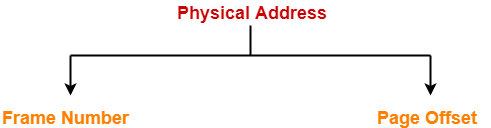
**Step-02:**

For the page number generated by the CPU,

* **Page Table** provides the corresponding frame number (base address of the frame) where that page is stored in the main memory.

**Step-03:**

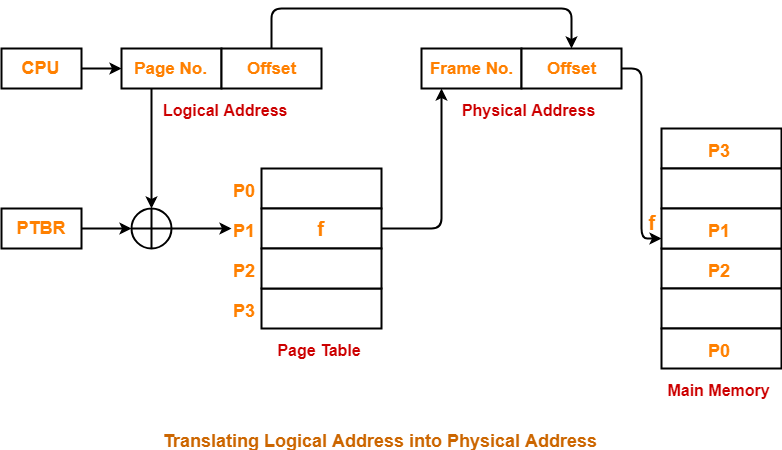
* The frame number combined with the page offset forms the required physical address.



* Frame number specifies the specific frame where the required page is stored.
* Page Offset specifies the specific word that has to be read from that page.

**Diagram-**

The following diagram illustrates the above steps of translating logical address into physical address-



**Advantages-**

The advantages of paging are-

* It allows to store parts of a single process in a non-contiguous fashion.
* It solves the problem of external fragmentation.

**Disadvantages-**

The disadvantages of paging are-

* It suffers from internal fragmentation.
* There is an overhead of maintaining a page table for each process.
* The time taken to fetch the instruction increases since now two memory accesses are required.

**Paging in OS | Formulas | Practice Problems**

**Paging in OS-**

**Important Formulas-**

The following list of formulas is very useful for solving the numerical problems based on paging.

**For Main Memory-**

* Physical Address Space = Size of main memory
* Size of main memory = Total number of frames x Page size
* Frame size = Page size
* If number of frames in main memory = 2X, then number of bits in frame number = X bits
* If Page size = 2X Bytes, then number of bits in page offset = X bits
* If size of main memory = 2X Bytes, then number of bits in physical address = X bits

**For Process-**

* Virtual Address Space = Size of process
* Number of pages the process is divided = Process size / Page size
* If process size = 2X bytes, then number of bits in virtual address space = X bits

**For Page Table-**

* Size of page table = Number of entries in page table x Page table entry size
* Number of entries in pages table = Number of pages the process is divided
* Page table entry size = Number of bits in frame number + Number of bits used for optional fields if any

**NOTE-**

* In general, if the given address consists of ‘n’ bits, then using ‘n’ bits, 2n locations are possible.
* Then, size of memory = 2n x Size of one location.
* If the memory is byte-addressable, then size of one location = 1 byte.
* Thus, size of memory = 2n bytes.
* If the memory is word-addressable where 1 word = m bytes, then size of one location = m bytes.
* Thus, size of memory = 2n x m bytes.

**PRACTICE PROBLEMS BASED ON PAGING AND PAGE TABLE-**

**Problem-01:**

Calculate the size of memory if its address consists of 22 bits and the memory is 2-byte addressable.

**Solution-**

We have-

* Number of locations possible with 22 bits = 222 locations
* It is given that the size of one location = 2 bytes

Thus, Size of memory

= 222 x 2 bytes

= 223 bytes

= 8 MB

**Problem-02:**

Calculate the number of bits required in the address for memory having size of 16 GB. Assume the memory is 4-byte addressable.

**Solution-**

Let ‘n’ number of bits are required. Then, Size of memory = 2n x 4 bytes.

Since, the given memory has size of 16 GB, so we have-

2n x 4 bytes = 16 GB

2n x 4 = 16 G

2n x 22 = 234

2n = 232

∴ n = 32 bits

**Problem-03:**

Consider a system with byte-addressable memory, 32 bit logical addresses, 4 kilobyte page size and page table entries of 4 bytes each. The size of the page table in the system in megabytes is \_\_\_\_\_.

1. 2
2. 4
3. 8
4. 16

**Solution-**

Given-

* Number of bits in logical address = 32 bits
* Page size = 4KB
* Page table entry size = 4 bytes

**Process Size-**

Number of bits in logical address = 32 bits

Thus,

Process size

= 232 B

= 4 GB

**Number of Entries in Page Table-**

Number of pages the process is divided

= Process size / Page size

= 4 GB / 4 KB

= 220 pages

Thus,

Number of entries in page table = 220 entries

**Page Table Size-**

Page table size

= Number of entries in page table x Page table entry size

= 220 x 4 bytes

= 4 MB

Thus, Option (B) is correct.

**Problem-04:**

Consider a machine with 64 MB physical memory and a 32 bit virtual address space. If the page size is 4 KB, what is the approximate size of the page table?

1. 16 MB
2. 8 MB
3. 2 MB
4. 24 MB

**Solution-**

Given-

* Size of main memory = 64 MB
* Number of bits in virtual address space = 32 bits
* Page size = 4 KB

We will consider that the memory is byte addressable.

**Number of Bits in Physical Address-**

Size of main memory

= 64 MB

= 226 B

Thus, Number of bits in physical address = 26 bits

**Number of Frames in Main Memory-**

Number of frames in main memory

= Size of main memory / Frame size

= 64 MB / 4 KB

= 226 B / 212 B

= 214

Thus, Number of bits in frame number = 14 bits

**Number of Bits in Page Offset-**

We have,

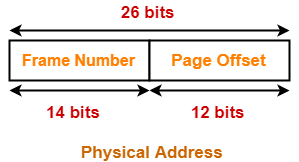
Page size

= 4 KB

= 212 B

Thus, Number of bits in page offset = 12 bits

So, Physical address is-



**Process Size-**

Number of bits in virtual address space = 32 bits

Thus,

Process size

= 232 B

= 4 GB

**Number of Entries in Page Table-**

Number of pages the process is divided

= Process size / Page size

= 4 GB / 4 KB

= 220 pages

Thus, Number of entries in page table = 220 entries

**Page Table Size-**

Page table size

= Number of entries in page table x Page table entry size

= Number of entries in page table x Number of bits in frame number

= 220 x 14 bits

= 220 x 16 bits (Approximating 14 bits ≈ 16 bits)

= 220 x 2 bytes

= 2 MB

Thus, Option (C) is correct.

**Problem-05:**

In a virtual memory system, size of virtual address is 32-bit, size of physical address is 30-bit, page size is 4 Kbyte and size of each page table entry is 32-bit. The main memory is byte addressable. Which one of the following is the maximum number of bits that can be used for storing protection and other information in each page table entry?

1. 2
2. 10
3. 12
4. 14

**Solution-**

Given-

* Number of bits in virtual address = 32 bits
* Number of bits in physical address = 30 bits
* Page size = 4 KB
* Page table entry size = 32 bits

**Size of Main Memory-**

Number of bits in physical address = 30 bits

Thus,

Size of main memory

= 230 B

= 1 GB

**Number of Frames in Main Memory-**

Number of frames in main memory

= Size of main memory / Frame size

= 1 GB / 4 KB

= 230 B / 212 B

= 218

Thus, Number of bits in frame number = 18 bits

**Number of Bits used for Storing other Information-**

Maximum number of bits that can be used for storing protection and other information

= Page table entry size – Number of bits in frame number

= 32 bits – 18 bits

= 14 bits

Thus, Option (D) is correct.