Chapter IV: Main memory management

I/ Generalities:

A memory is a system capable of acquiring, storing and retrieving binary information in a computer. The capacity of a memory is the amount of information it can store, expressed in bits or words of 2n bits.

Access time is the time that elapses between the request for information and the moment when it becomes available.

Four types of memory differ according to use, capacity, access time and cost:

- Registers (these are the smallest memories, have the shortest access time and are the most expensive).
- Cache memory (low capacity, very fast, very expensive).
- Main memory (128 MB capacity → 2 GO, very fast access, less expensive).
- Auxiliary memory (very high capacity, long access, least expensive).

II/ Main memory definition:

The main memory is a set of memory locations or words, each with its own address, which responds to read and write operations. The MM is used to store data and programs during execution, and is the intermediary between the processor and the rest of the computer's components.

The set of programs used to manage and control the use of main memory is known as the main memory management system.

III/ Memory manager objectives:

The objectives of the MM manager are :

1/ To share and control memory space between processes.

2/ Protect the space allocated to each process.

3/ Reallocate and rearrange memory to optimize its use.

4/ Logical organization, i.e. logical division of space into partitions, pages or segments.

To achieve this, the memory manager:

- Allocates memory space to processes.
- Frees occupied space.
- Keeps track of occupied and free memory.
- Manages swapping operations between MM and secondary memory.

IV/ Different ways of sharing the MM:

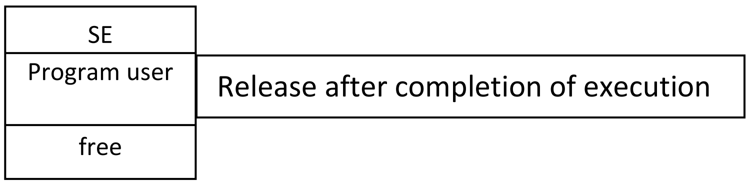
1/ **Modes not using virtual memory** :

Programs are loaded only into the MM:

a/ Single contiguous zone: used in sequential systems, where only one program exists at a time, so the MC is shared between this program and the operating system.

Disadvantages :

- Poor memory usage.
- Program sizes are limited to this area.



Note: This scheme was used in the first minicomputers (IBM. DS/360).

b/ Multi-partitioning scheme: applied when running several programs (multiprogramming) ⍰ is to divide space into n partitions:

➢ Multiple fixed partitions:

The MC is divided into n partitions of fixed sizes when the OS is generated, and this partitioning remains unchanged until the machine is shut down. An allocation algorithm is responsible for finding a sufficient size for the pending program.

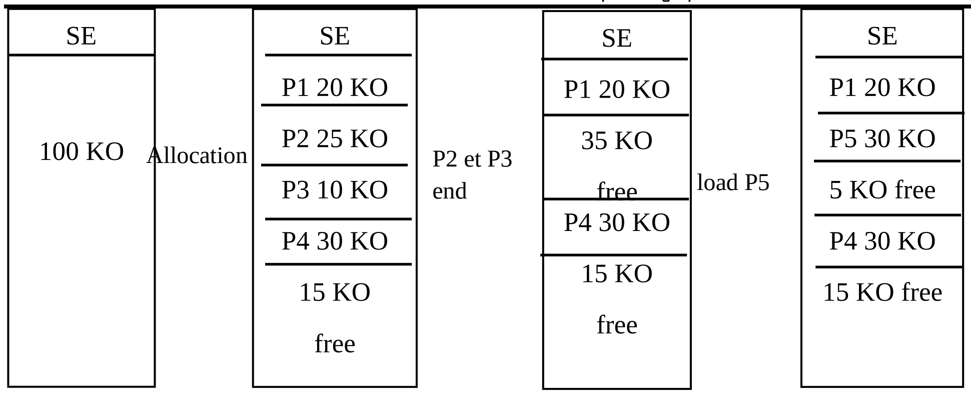
Disadvantages :

- Appearance of several fragments in unusable memory.
- Partitions may be free while programs are waiting ➝ program size > partition size.
- Problem with absolute programs requiring absolute loading addresses ➝ where free partitions cannot be used.

➢ Multiple variable partitions:

Memory is dynamically partitioned according to program demand.

Example: 100KO partition for user program, P1 = 20, P2 = 25, P3 = 10, P4 = 30, P5 = 30



Allocation in this strategy is made according to one of the following algorithms:

- **First Fit**: The program is placed in the first partition found >= that of the program. This has the disadvantage of leaving several fragments unusable.
- **Best Fit**: this strategy seeks to find the best partition whose residual is minimal (partition size - program size <<<).
- **Worst Fit**: the program is placed in the largest partition, with the aim of gaining a new partition for another program.

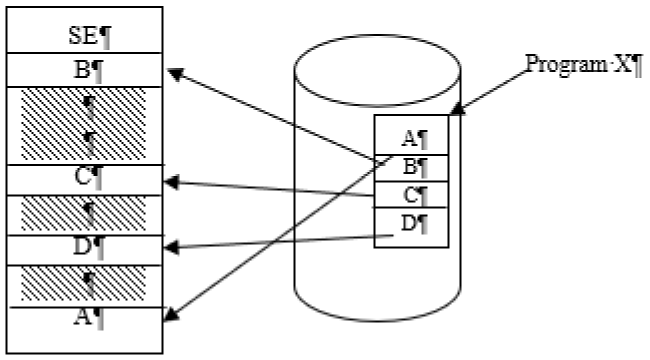
MM compaction: to avoid unusable fragments created during process allocation, a compaction algorithm enables free partitions to be combined into a single partition with a large size. To perform this operation, the programs to be moved must be relocatable.

2/ Modes using virtual memory:

a/ Virtual memory :

The principle of virtual memory is to stretch the physical space of the MC to a larger memory space, using part of the secondary memory, exactly the hard disk. Programs are scattered between the MC and the MS.

Programs are loaded into the MS at the start of launch, then cut into blocks and brought back from the MS block by block to the MC, with the execution time of each block.



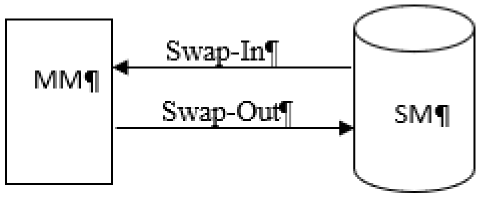
Virtual memory is used for the following purposes:

- Executing programs when they are partially in MC.
- Loading and execution of programs that are larger than in MC.
- Increase the degree of multi-programming.

b/ Swapping:

This is an operation used in the case of virtual memory, such as :

- A program is loaded in its entirety into MS, then transferred part by part into MC (Swap-In).
- It runs until either termination or blocking.
- In the case of blocking, the program can be transferred to MS (Swap-out).



To manage the space in these two memories and to facilitate swapping operations, the system uses one of the following techniques:

- Pagination
- Segmentation
- Paginated segmentation

c/ Pagination:

Pagination is used when :

- Main memory is organized into fixed blocks of equal size, called **physical pages or page frames**.
- Processes are divided into blocks called **virtual pages** of the same size as the page frames.

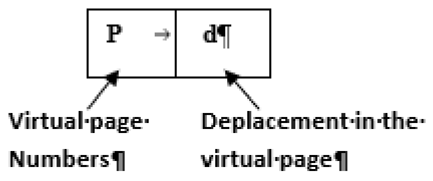
Programs are then stored in secondary memory in a swap area, where a virtual address space is manipulated. A program is executed by bringing back from MS to MC, virtual page by virtual page, the moment it is referenced in a free page frame.

Where the processor generates the address of the next instruction, which is a virtual address whose page containing this address must be referenced.

✓ Correspondence between virtual and physical addresses :

When virtual memory is used, the referenced address is not put directly on the address bus, but is sent to a physical unit called the Memory Management Unit (MMU), whose role is to map this address to its real address in the MC, such as :

- The processor generates a virtual address.
- The MMU transforms this address into a :



- The memory manager consults the page table of the running program to find out whether this page exists in the MM.
- If this page does not exist in the MM, it signals a page fault, and loads it from the SM to the MM.
- It maps the virtual move to the instruction's physical address in the MM,
- Now this physical address is transferred to the address bus.

Page table: this table contains all the information needed to manipulate pages, and is located in the MC included in the Process Control Block:

- Page number.
- The address in the MS → S.
- A bit indicating its existence in the MM → R.
- The address of this page if it is loaded into the MM → PR.
- A bit indicating whether the page is modified or not → M.
- A protection bit.
- A reference bit.

✓ Virtual memory management in pagination:

1/Research: the manager looks for the existence of the virtual page to load it into the MM, either in the processor's request, or by anticipation (usage prediction).

2/Placement: where the page should be placed in the MM.

3/Movement: which page should leave the MM, if it is saturated, to load a new referenced page, the manager looks for a victim page to replace according to one of the following strategies:

- **FIFO strategy** → the oldest page is replaced.
- **LRU strategy** → Least Recently Used.

- **Optimal algorithm** →prediction of the page that will not be referenced for the longest time).
- **NRU strategy** → Not Recently Used.
- **Second Chance** → FIFO enhancement, the first entry + it is never referenced.

✓ Advantages and disadvantages of pagination

Advantages :

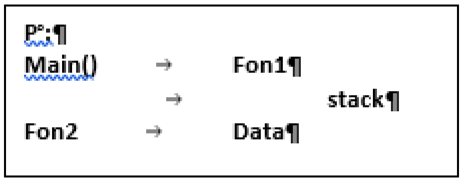
- Easy loading of virtual pages into physical pages.
- Simple correspondence between physical and virtual addresses.
- The programmer does not need to know the technique used by the manager.

Disadvantages :

- Increase in page table size relative to program size.
- High rate of page faults.
- Data and procedures are not separated, making them difficult to share between different programs.
- Data and procedures are not protected simply and separately.

d/ Segmentation:

A strategy that reproduces memory partitioning as described by the user and the compiler (logical partitioning).



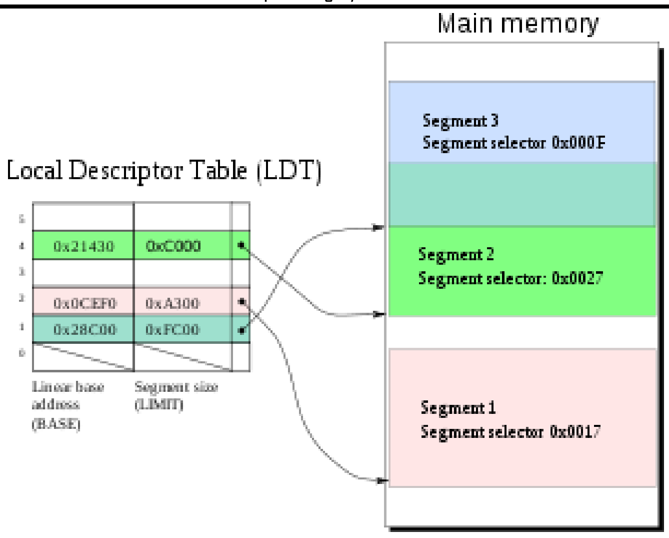
```
P:¶
Main()   →   Fon1¶
          →          stack¶
Fon2     →   Data¶
```

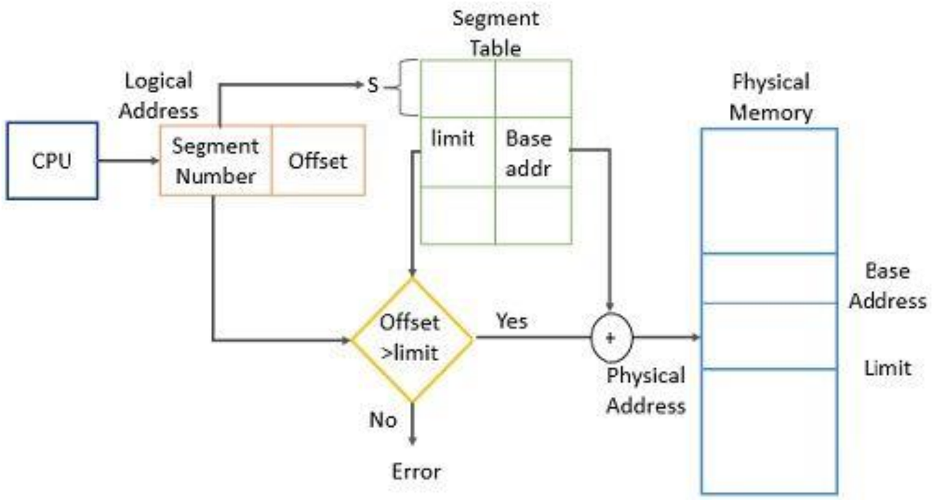
Where:

Central memory is organized into blocks of variable size → **Segments physical**.

Each process is divided into logical units → Virtual segments (global variables, procedure call stack, each procedure or function, local variables, main program).

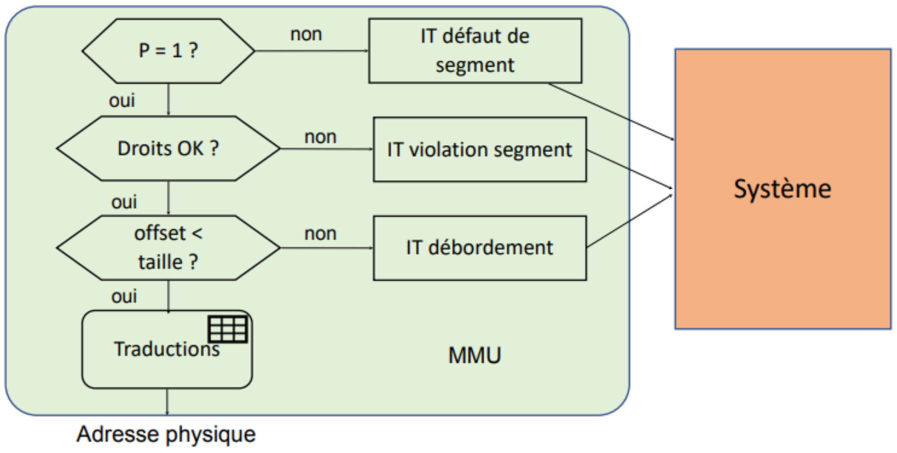
Note.: the segments making up a program are constructed during compilation.





A segment table entry (P, Rights, Size, Base)

- P: presence (0: not present, 1: present) ;
- Rights: r (Read), w (Write), x (eXecutable). - The MMU (processor) checks for an address



Example of segmentation in 8086 architecture:

The 8086's addressable memory space is 220 = 1,048,576 bytes = 1 MB (**20 address bits**). This space is divided into segments. A segment is a 64 KB (65,536 bytes) memory area defined by its starting address, which must be a multiple of 16.

In such an address, the 4 least significant bits are set to zero. We can therefore represent a segment address with only its 16 most significant bits, the 4 least significant bits being implicitly set to 0.

To designate one of the 216 = 65,536 memory locations contained in a segment, all you need is a 16-bit value.

Thus, a memory location is identified by the 8086 using two 16-bit quantities:

- a segment address;
- a displacement or offset (also called effective address) within this segment.

➞ This method of memory management is called memory segmentation.

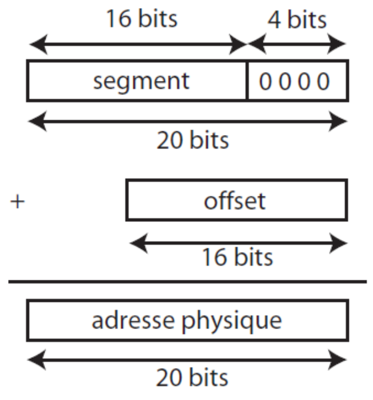
A (**segment,offset**) pair defines a logical address, expressed as ➞ **segment : offset**

The address of a memory cell given as a 20-bit quantity (5 hex digits) is called the physical address, as it corresponds to the value actually sent on the A0 - A19 address bus.

At any given moment, the 8086 has access to 4 segments whose addresses are in the segment registers CS, DS, SS and ES. The code segment contains the program instructions, the data segment contains the data manipulated by the program, the stack segment contains the backup stack, and the additional segment may also contain data.

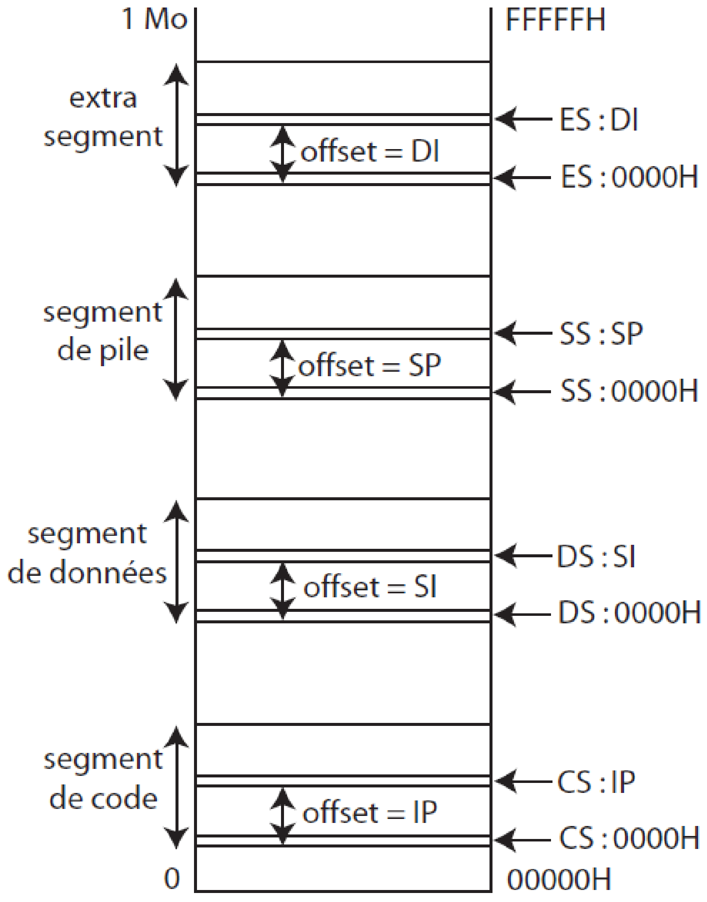
The CS register is associated with the IP instruction pointer, so the next instruction to be executed is at the logical address CS : IP.

The physical address for the location of the instruction is generated by shifting the CS left one hex digit and then adding it to the IP. IP contains the offset address.



physical address = 16 X segment + offset

Similarly, the segment registers DS and ES can be associated with an index register. Example: DS: SI, ES: DI. The stack segment register can be associated with pointer registers: SS: SP or SS: BP.



Note: in other systems, memory is organized into pages - Memory pagination.

e/ Paginated segmentation:

The aim of this technique is to combine the advantages of the two previous techniques:

- The program is first split into logical segments of variable size, each segment being split into virtual pages of the same size.
- The main memory is divided into pages of the same size.
- The virtual address is a triple consisting of the segment number, page number and page displacement.

V/ Memory protection:

Every system must offer a mechanism to protect against unauthorized access to memory blocks, due to programming or other errors. Processes must be protected from each other's activities. Two modes are available:

1/ All-or-nothing protection:

Access to a memory area for a given process is either completely authorized or forbidden, i.e. by knowing the start and end limits of this area, we indicate whether what exists between these limits is protected or not.

2/ specific protection:

Allows you to specify the operations allowed on a memory area:

- Read.
- Write.
- Execute.

Three RWE protection bits are specified, with different values depending on the level of protection (Example: 000 ⍰ access denied, 001 → execution only, ...).