

A1:

* **Dezavantaj**:

1. There cannot exists more than one process at a time

* **Avantaj:**

1. Simplicity in implementation

A2:

* **Dezavantaje:**

1. The number of processes that are in execution is limited to the number of partitions of the memory
2. You cannot run simultaniously two programs that are compiled for the same partion only one at a time
3. You are limited for the size of the program by the size of the partitions.

* **Avantaje:**

1. It works directly with memory adresses so it is very fast

A1,A2 exec adress=physical adress

A3:

Exec adress becomes an offset physical adress = partition start + offset

* **Dezavantaje**:

1. The number of processes that are in execution is limited to the number of partition of the memory
2. You are limited for the size of the program by the size of the partitions

* **Avantaje**:

1. It solved the colision pb of 2 programs which are trying tor un in the same partition

A4:

* **Dezavantage**:

1. We may have memory fragmentation . The only sol for this is coalescing free memory but that is very slow because we need to shift all the processes in memory or coales until enough ( only shift processes so we get enough space in order to run the program ) but we have to do this more times .

* **Avantage**:

1. We have no limits in size and in number of procceses
2. (Intre A4 si oricare de dupa) the physical adress calculation is simpler and takes less time

MEMORY FRAGMENTATION: paged allocation

A5:

* **Avantaj**:

1. Because we save the programs by pages of 4kb we soved the pb of MEMORY FRAGMENTATION

* **Dezavantaj**:

1. MEMORY CALCULATION we need to look up each time at the memory adress in the table

A6:

* **Avantaje**:

1. Access more memory than address size allowed
2. Prevent programs from overwritting their own code
3. i.e if you have one library you can simply reuse it, not write it again. (Reusable segment , save some memory )

* **Dezavantaje**:

1. Threre is the pb of memory segmentation

A7:

* **Avantage**:

1. We have more memory and we solved the pb of memory fragmentation.

* **Dezavantaje**:

1. The computation of the address is even more complex because now the virtual address is a triplet of segment, page and offset
   * Load all program pages into RAM from the biginning:

* Avantaj:

1. fast execution

* Dezavantaj:

1. slow startup
2. Occupy memory with data that is never used
   * Load the first page and then load every page when needed

* Advantage:

1. No wasted memory

* Disadvantage:

1. Slower execution
   * Locality principle (a process is likely to need soon the pages next to the page that was just loaded:

SOLUTION: prefetch neighbouring pages

* Avantage:

1. Combination of fast execution and no wasted memory

* Disadvantage:

1. If the locality principle is not respected we will prefetch neighbouring pages without being needed

3.

FIFO- rather random

NRU (not recently used) : mark every page with 2 bits r,w that are set to 1 when a r/w operation is done on the page and are periodically reset to 0.

0 = 00; 1 = 01 (r, not w) ; 2 =10 ; 3=11

* Advantage :

1. Simple algorithm , more efficient than FIFO

* Dezavantaj:

1. When there are no more programs in the first 3 cathegories (0,1,2)we need to stop a process that has been recentely read and written.

LRU (least recently used)

Small and quick

For a system with N pages maintain an Nx N matrix pf bits, as follows:

* When a page is accessed, fill its line with 1 and its colomn with 0
* Choose a victim page from those with the minimum line sum

1. MALLOC
   * First-Fit:

* Advantage:

1. fast

* Disadvantage:

1. Does not adress fragmentation
   * Best Fit:

* Disadvantage:

1. Slower
2. Creates very small free slices -> memory fragmentation
   * Worst Fit

* Advantage:

1. Leaves large free slices

* Disadvantage:

1. Slower
   * Buddy Allocation

* Faster, fragmentation da
* Slower fisrt fit faster than... . More complicated to implement