INTERNATIONAL JOURNAL OF RESEARCH IN COMPUTER APPLICATIONS AND ROBOTICS
ISSN 2320-7345

MEMORY MANAGEMENT ALGORITHMS REPRESENTATION TO CONVERT LOGICAL ADDRESS TO REAL ADDRESS

Maha A. Sayal

Computer Science Department, University of Thi-Qar, Thi-Qar, Iraq
maha.asyal@utq.edu.iq

Abstract: - The main aim of computer system is to execute programs. These programs must be at least part of them in the main memory during execution. To improve both user responsiveness and CPU usage, the operating system must have diagrams and algorithms to administrate the memory and the different response of each algorithm according to the conditions. This depends on many factors, including hardware support and design requirements for algorithms [1].

The aim of the paper is to represent how the main memory is managed by the operating system, to clarify the work of the algorithms used in memory, and how to convert the logical address generated by the central processing unit to a real address that the memory can read. This is done by programming the algorithms that is used in memory management and convert the logical address to a real address using visual Basic language.

Algorithms have been tested by this paper based on the issues in the curriculum. This work is part of an educational package used to illustrate the operations of the operating system and its proposal to represent the practical part of the operating system for the fourth phase.

Keywords*:* Main memory; Memory management; Logical address; Physical address.

1. INTRODUCTION

The management of the computer resources is one of the functions of the operating system. The most important of which is the main memory, because it is the only place where the CPU calls the program and data instructions to be implemented. The part of the operating system that is responsible for more task in the memory is called memory manager, the most important tasks are: Monitor the status of all memory locations in terms of emptiness in order to immobilize the processes to be executed or completed in order to dump sites after the completion of operations from the implementation. Define the way that is distributing empty sites for the operations to be performed .and specifies priorities in immobilizing. And transfer operations performed from main memory to secondary memory (hard drive) or vice versa [2].

In management and organization of the main memory there are options, the first include the first place and how the distribution of empty sites in memory is allocated to one operation or distributed to several operations,

the second option are divided sites equally for operations or divided according to the need of the process, all this according to the algorithms used in management [3].

2. PROPOSED WORK

In this paper, the following memory management algorithms and their work will be represented as a program. But in start, the work of algorithms must be explained:

2.1 Fixed Partitioning Algorithm

It is the division of memory into specific fixed partitions (small, medium, large), each operation assigned to the appropriate partition. But there may be part of partition or whole partition that are not allocated, because there is no process within the range of there sizes, although there is a set of operations in the waiting queue. This is called internal and external fragmentation [4], as in Figure (1).

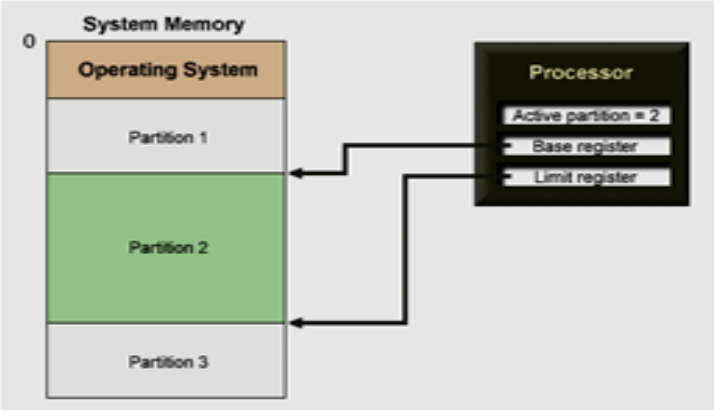

Figure 1: Fixed Partition

2.2 Dynamic Partition Algorithm (First Fit Strategy)

This strategy is based on the principle of settling the process in the first sector that is not busy in the memory of the process, regardless of the size of the sector [5], as in Figure (2).

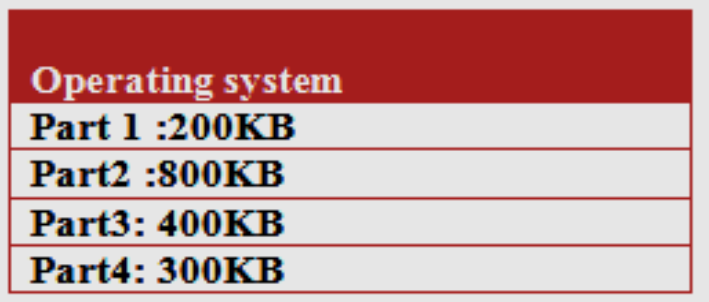

Figure 2: Partition variable

2.3 Segmentation algorithm

Segmentation algorithm is a memory management scheme that supports the user's view of memory, the main goal of which is to reduce the space. As far as possible here is the optimal monopoly of memory fragmentation [6]. As in Figure (3).

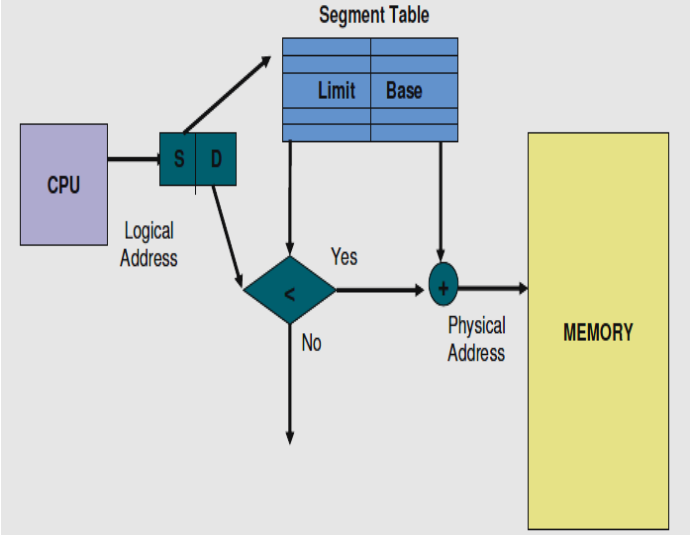

Figure 3: Segmentation

This algorithm converts logical address to real address*:*

$V=(s, d).....1,$ V virtual address or called logical address, s means segment number and d means offset or displacement.

$R=s'+d \ .....2,$ R real address or called physical address, s' converting s after segment table, d is offset or displacement [4].

2.4 Paging algorithm

Paging algorithm is a memory management scheme that allows the real address space of the process to be non-bypassed so as to avoid external fragmentation and the need for compaction. And the use of memory with best way by dividing memory into equal parts [7], as in Figure (4).

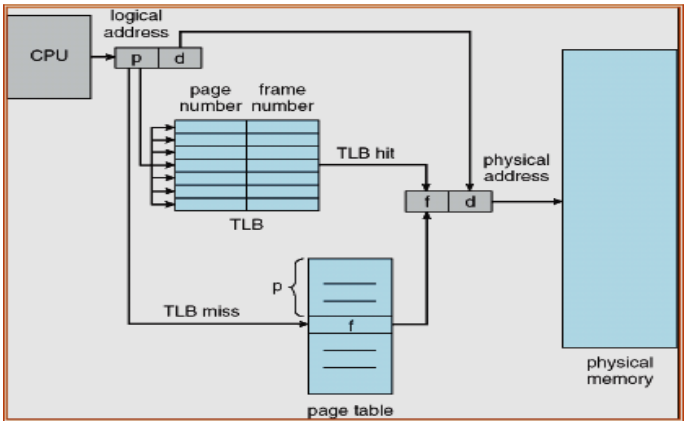

Figure 4: Paging algorithm

This algorithm converts logical address to real address*:*

$V= (p, d).....3,$ V virtual address or called logical address, p means page number and d means offset.

R=p'* page size + d .....4, R real address or called physical address, p' converting p after page table, d is offset [4].

3. METHODOLOGY

The paper is mentioned how to create the proposed interface and to program the required algorithms using the Visual Basic language.

3.1 Tools used

Visual Basic is a powerful and effective tool for developing applications that are compatible with Windows environment. Provides an easy-to-use IDE where the screens and windows of the program are designed with mouse clicks and animations to create an interface that fits the job programmer wants [8].

3.2 Programming algorithms

First, certain tools are chosen for representing algorithms. Name of tools must be changed according to the algorithms. In the fixed partition and Variable partition algorithms, command, four texts and four labels must be added. While paging algorithm must be have command, twelve texts and three labels. And Segmentation algorithm needs command, twelve labels and sixteen texts, as shown in figure (5).

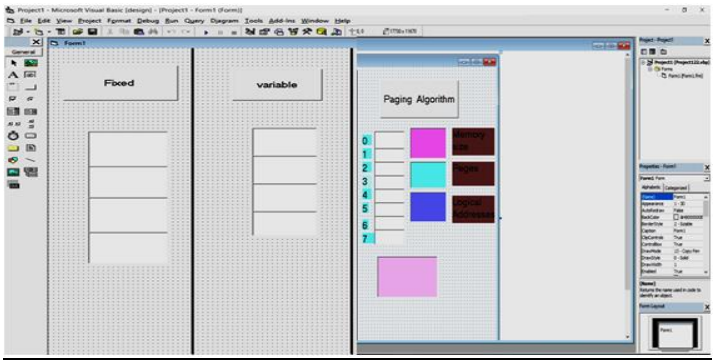

Figure 5: Used tools

The code for each algorithm is written in the custom command as shown in Figure (6).

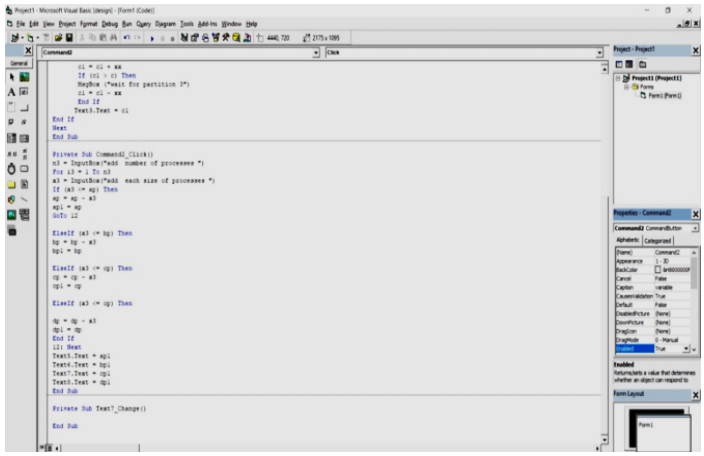

Figure 6: Writing the code (fixed)

Another command must be added, given name to it called (exit) and placed the closing code to end the program, so the final interface is as follows in figure (7).

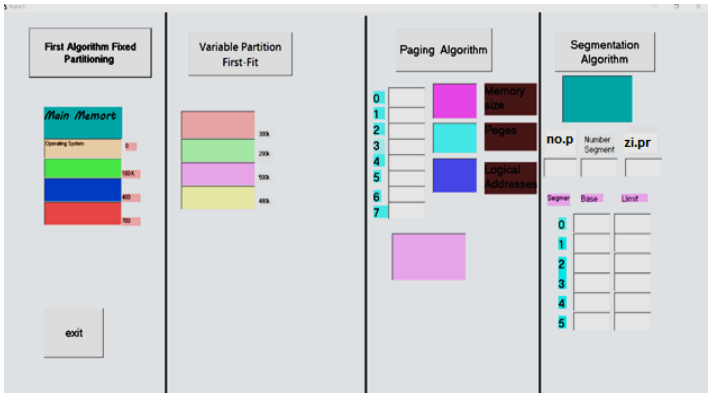

Figure 7: Final interface

4. RESULTS

Algorithms were tested using some of the examples in the curriculum for operating systems**.**

4.1 Test the fixed partitioning algorithm

Test example: Four processes must be added to main memory using fixed partitioning algorithm.

P1=80, P2=70, P3=120, P4=240.

Solution:

Number of processes must be entered as shown in figure (8).

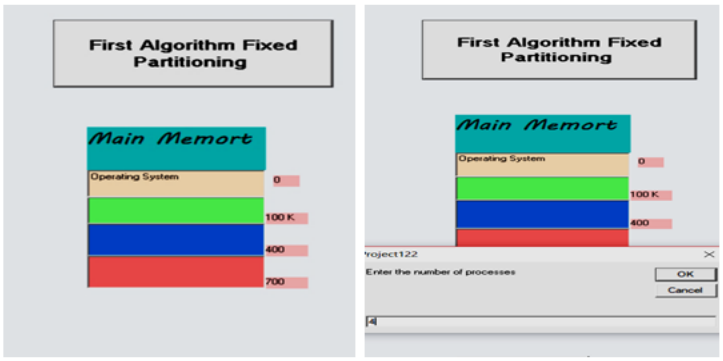

Figure 8: Interface fixed partition & Entering No. Pr.

In this algorithm, process will look for partitions and compare between size of partitions and size of process. If size of process within range of any partitions (small, medium or large), process will be put in appropriate partition. If appropriate partition is filled or stayed part of partition is not fit, the process must be waited. As shown in Figures (9),(10),(11),(12).

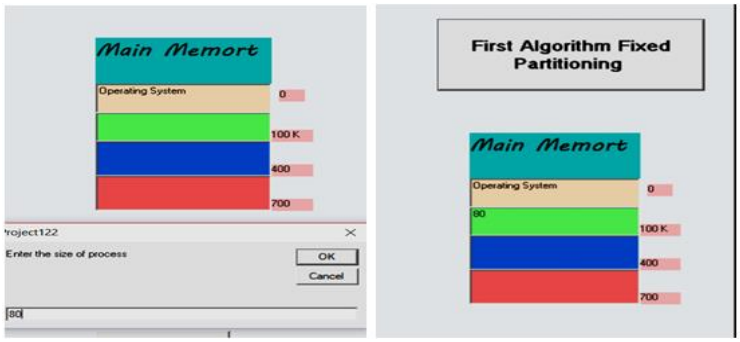

Figure 9: Inserting and executing the first operation size

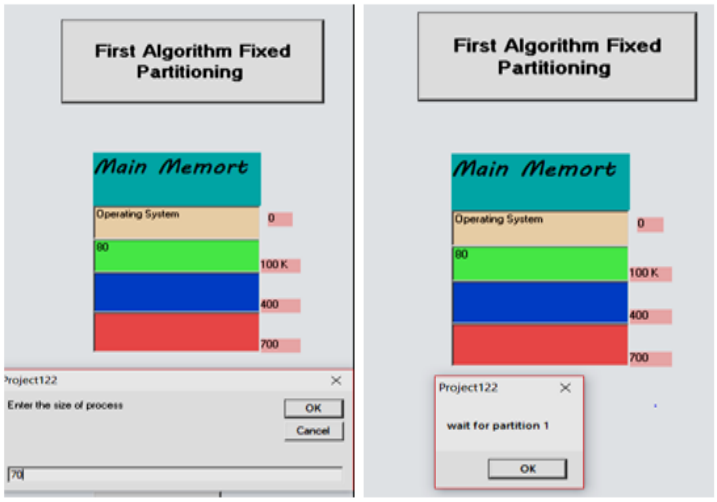

Figure 10: Inserting and executing the second operation size

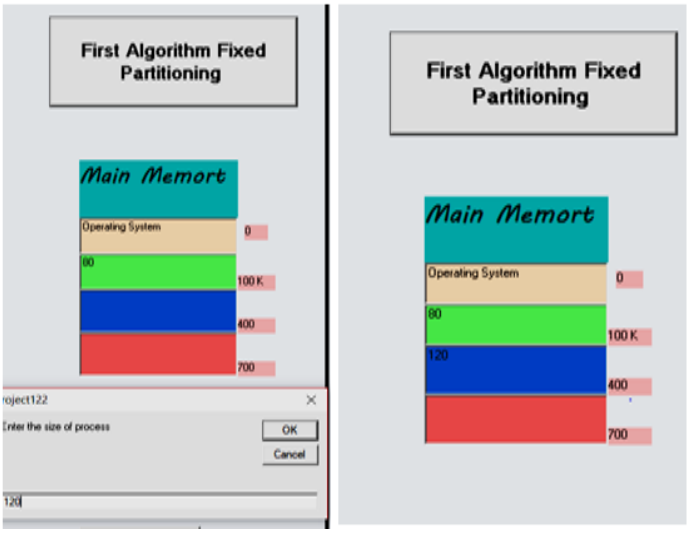

Figure 11: Inserting and executing the third operation size

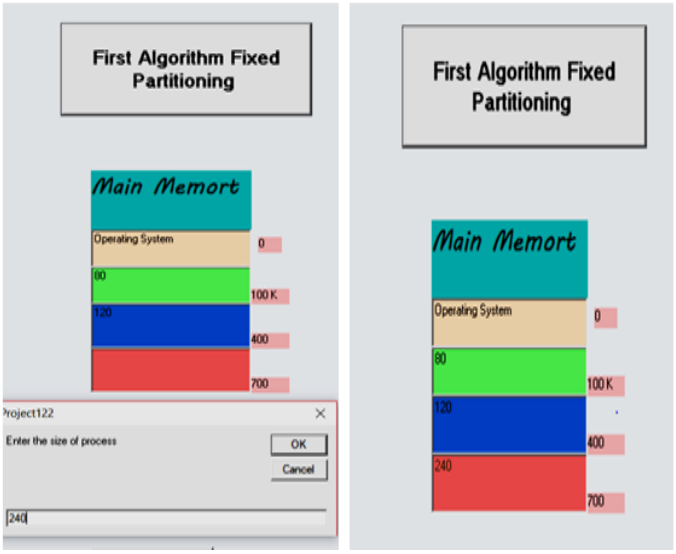

Figure 12: Inserting and executing the fourth operation size

4.2 Test the variable partitioning algorithm(First Fit Strategies)

Test example: Four processes must be added to main memory using variable partitioning algorithm. P1= 200, P2=400, P3=100, P4=450.

Solution:

In this algorithm, number of processes must be specified, as shown in figure (13).

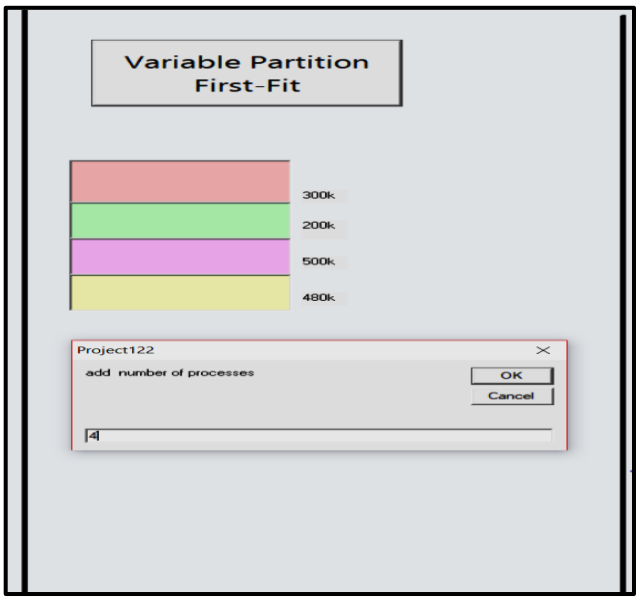

Figure 13: Interning number of processes

process will look for the first appropriate blank from beginning of memory and it must be put in this blank .the second process will also look for the new first blank to put the process. As shown in Figures (14),(15),(16),(17).

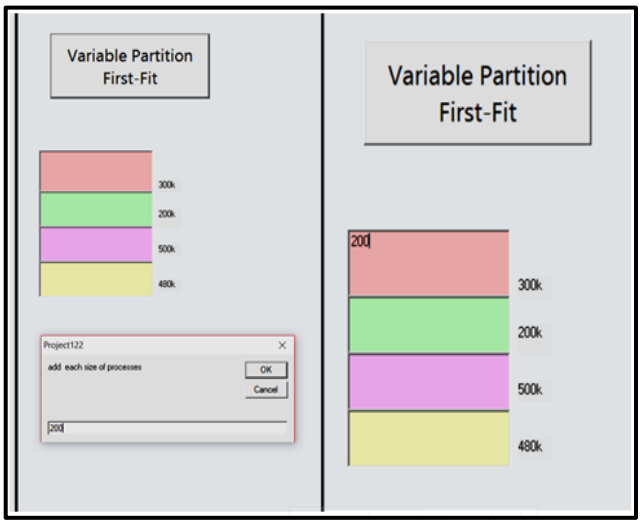

Figure 14: Inserting and executing the first operation size

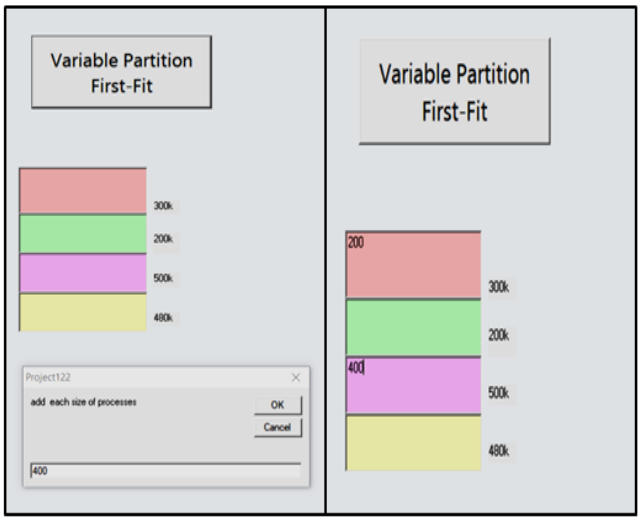

Figure 15: Inserting and executing the second operation size

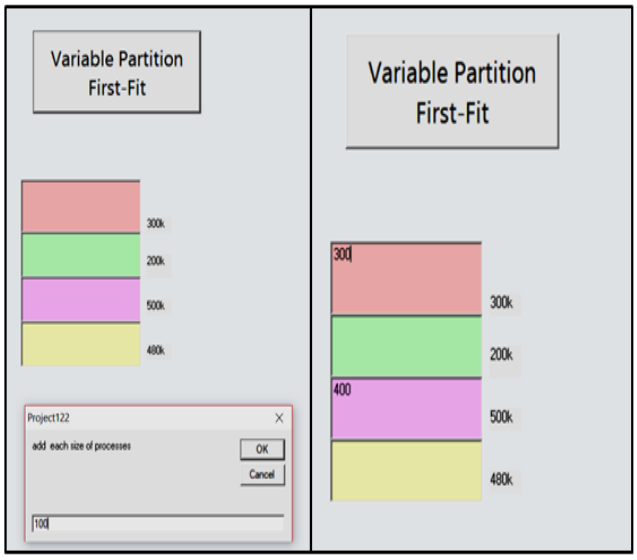

Figure 16: Inserting and executing the third operation size

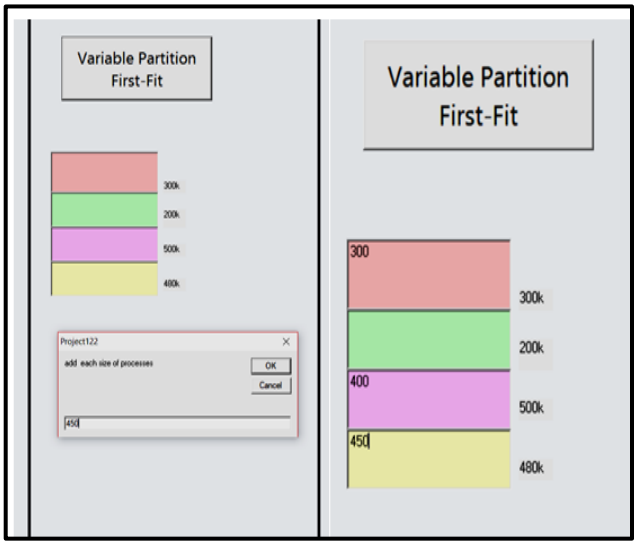

Figure 17: Inserting and executing the fourth operation size

4.3 Test the Segmentation algorithm

Test Example: What is the real address of the following logical address?

a. <0,430> b.<1,11>

Note: Figure (18) is the table of segmentation in the system & segmentation algorithm.

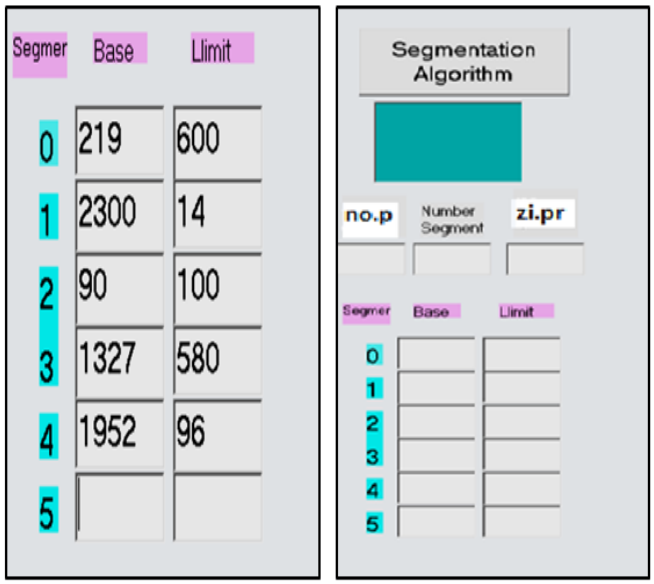

Figure 18: Segment table & Segmentation algorithm

Solution

In segmentation algorithms, logical address consist of two parts < **s, d** > **s** means segment number and **d** means displacement or offset.

First, segment number must be found in segment table above and observed **base** and **limit** in same row of segment table, then compare limit with displacement.

If limit > d then

Physical address = base + limit

Else

Error in addressing mode

(a) <0,430> means the logical address. 430 is displacement , 0 is segment number , that mean base is 219 and limit is 600 according to segment table, limit and displacement must be compared, 600 > 430 then physical address = base + limit , physical address = 219 + 600 = , physical address =819, as shown in figure 19 Perform (a).

(b) <1, 11> means the logical address. 11 is displacement , 1 is segment number , that mean base is 2300 and limit is 14 according to segment table, limit and displacement must be compared, 14 > 11 then physical address = base + limit , physical address = 2300 + 14 = , physical address =2314 , as shown in figure 19 Perform (b).

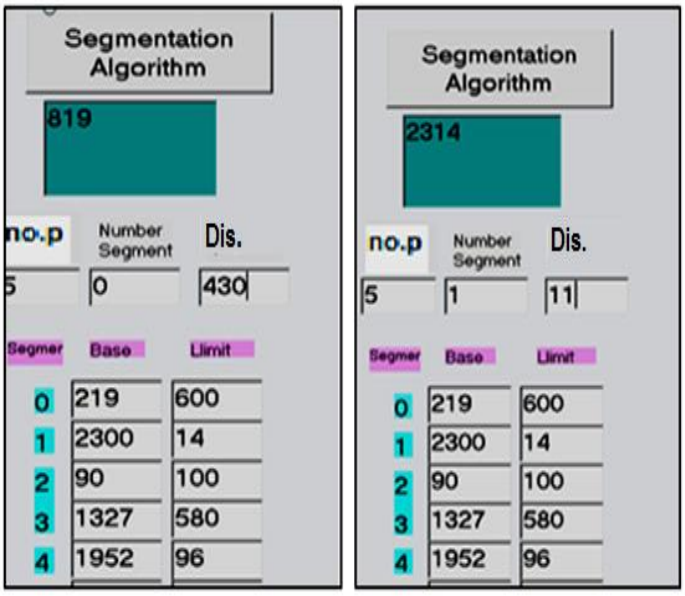

Figure 19: Perform (a) & Perform (b)

4.4 Testing the Paging algorithm

Test Example: Memory managed by the 512-byte paging algorithm with 16 frames. Its logical address consists of 8 pages and the size of each page is 32 bytes as shown in table.1, what is the real address of the following logical addresses? a- 250 b- 78.

Table 1: Paging table

page	frame
0	12
1	4
2	6
3	11
4	8
5	7
6	3
7	10

Solution:

(a) 250 is a logical address , page size = memory size / frames = 512/16, page size = 32 bytes, to obtain real address , page number & offset must be extracted ,

Page number =logical address div page size , page number =250 div 32, page number=7.

Page number is 7; therefore, frame number is 10 according to paging table above.

Offset = logical address mod page size, offset = 250 mod 32, offset =26.

Physical address = frame number * page size + offset

Physical address= 10 *32 +26

Physical address= 346, as shown in figure 22.

(b) 78 is a logical address, page size = memory size / frames = 512/16, page size = 32 bytes, to obtain real address, page number & offset must be extracted,

Page number =logical address div page size , page number =78 div 32, page number=2.

Page number is 2; therefore, frame number is 6 according to paging table above.

Offset = logical address mod page size, offset = 78 mod 32, offset= 14.

Physical address = frame number* page size + offset

Physical address= 6 *32 +14

Physical address= 206, as shown in figure 23.

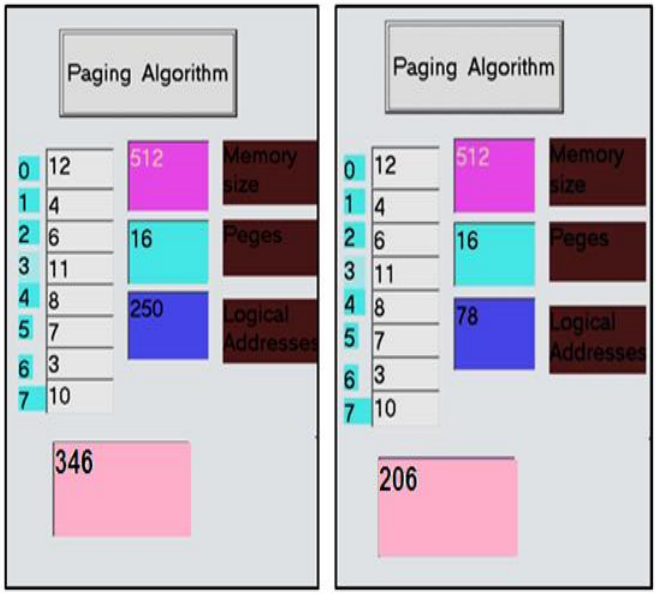

Figure 20: Perform (a) & Perform (b)

5. Conclusion

✓ The representation of memory management as a program shows simply how converting the logical address through the stages of the translation of the address from the segment number and offset (generated by CPU) to the numbers (stored in the main memory).
✓ The possibility of the adoption of the program as an educational package given to the material of operating systems as part of the practical side**.**

6. FUTURE WORK

✓ Adding an algorithm that combines Paging and Segmentation into a single algorithm.
✓ Adding method of how to convert the logical address to a real address by means of the registers associated with the main memory.
✓ Representing other operating system processes such as representing deadlock algorithms as status of system, requesting sources, and deadlock detection.

REFERENCES

[1] W. Stallings .*Operating Systems, Internals and Design Principles*, 7th edition, Prentice Hall, Pearson, 2012, pp.307-360.
[2] I.A. Dhotre. *Operating Systems.* First edition, Technical Publication pune, 2009, pp.1-28.
[3] G.Gagne ;A. Silberschatz and P.B.Galvin . *Operating Systems Concepts*.8th edition, John Wiley&Sons, Asia, 2012, pp.315-417.
[4] J. Breecher . "Section08-Memory_Management" in *Operating Systems*, 2009, pp1-35.
[5] R. Pandey, "Memory Management" ,Department of Computer Sciences, University of California, Davis , 2011.
[6] Poornima; C.V. Guru Rao; N. Thabassum and S.P. Anandaraj , " Memory Management by Using the Heap and the Stack in Java" , *International Journal of Research in Computer Science & Engineering*, Vol–4, Issue-6, Nov – Dec,2014 .
[7] Mishra, B.; Singh, N.; Singh, R. "Master-slave group based model for co-ordinator selection, an improvement of bully algorithm". International Conference on Parallel, Distributed and Grid Computing (PDGC). pp. 457–460,2014.
[8] G.Haggard; W.Hutchison and Ch.Shibata. *Introduction: Visual BASIC 6.0* .1st edition, 2013, pp.10-227.