# OPERATING SYSTEM

Lecture 3

## Paging and Segmentation

Dr Victoria Mukami

## INTRODUCTION

This lecture continues with memory management, and it features the paging memory allocation scheme. Specifically, it will do an overview of the scheme while reviewing the page replacement schemes. The two replacement schemes will include First In First Out (FIFO) and Least Recently Used (LRU). Finally, a review of segmentation will be done.

### Learning objectives

By the end of this topic, you should be able to:

1. Highlight the characteristics of paged memory segmentation
2. Explain the two types of page replacement methods
3. Show an understanding of segmentation

## OVERVIEW

During the last lecture, we reviewed the early memory management systems allocation schemes and systems. This week we look at paging systems. Operating systems have evolved to support multiprocessing and multiprogramming and newer memory management schemes have been used to ensure that efficiency and minimal wastage of the memory occurs. Older memory systems required an entire job or program to be loaded into memory. Newer systems do not require the entire program to be loaded into memory. Instead, newer systems only require the part of the program required for processing to be loaded into memory. We first review the paged memory allocation.

## PAGED MEMORY ALLOCATION

The best way to explain this memory allocation is by using a book. A book is made of pages. Each page could hold its content or could be a continuation of the previous content. One thing to understand is that a book can have a few pages or a lot of pages. Paged memory is like that. It divides jobs into units of equal size with each unit called a page [1].

Page size can be the same size as a section of main memory, and this is called a page frame [1]. The same is said for the sections of a magnetic disk called sectors that would be the same size as the page. Ideally, when the page, sector and page frame are all the same size then the paged memory is very efficient [1].

Paging deals with the issues of internal fragmentation from fixed partitions and external fragmentation due to dynamic partitions. Like the partitioning scheme, a list of free page frames is maintained by the operating system. Before any job can be executed, the operating system performs a couple of functions [1]:

- determine the number of pages within the program or job
- locate the empty page frames using the list of free page frames
- finally, load all the program's pages into the frames

During the initial loading, the program is prepared in sequence since instructions are found on the first page and the last page to signal the beginning and the end. The pages are not loaded contiguously into memory like the single-user system of early memory systems [1]. The pages are loaded anywhere there is space within the page frames in memory.

With this new feature where pages are loaded anywhere in memory, it results in the operating system having to figure out how to manage the non-contiguous pages. This is done by the Job Table (JT), Page Map Table (PMT) and the Memory Map Table (MMT). Before we review the tables, let us review through a diagram how the paging memory allocation works.

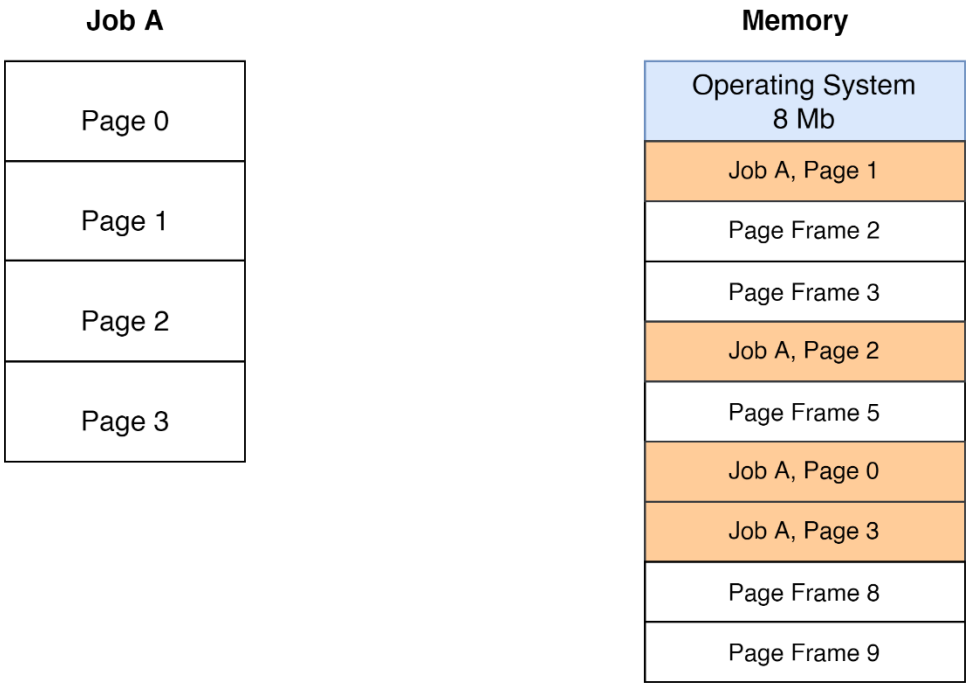


Figure 1: Paged Memory Allocation

As shown above, Job A is divided into pages 0 to 3. The first Page 0 and the last Page 3 contain the job instructions. The pages are divided into similar sizes as the page frames shown. These pages are then loaded into memory non-sequentially. For instance, page 0 is loaded into Page Frame 6 while Page 1 is loaded into Page Frame 1.

The Job Table (JT) is used to hold all the jobs within the system as well as the resources they need and their location within the PMT. Each job then has its own PMT. This lists the number of pages that the job has been split into. You will realize from our previous example Figure 1; the pages begin from 0 onwards. The memory map shows where each page is in memory and whether the memory section is free or busy [1].

**Advantage**

- Memory usage is more efficient as pages are stored anywhere there is space within the page frames

**Disadvantage**

- Since pages are located anywhere in memory, the operating system needs additional resources to keep track of each job's resources.
- The last page of any job will more often result in internal fragmentation. An example would be pages and page frames that are divided into 100Mb blocks. This means that a job that is 380Mb would be divided into 4 pages with the fourth page having only 80 Mb and thereby resulting in internal fragmentation.
- A small size page would result in a PMT that is lengthy meaning more resources are required by the operating system to manage the tables. Too large a page will mean that there will be larger internal fragmentation. Success for this memory allocation scheme lies within a determination of the page size.

**DEMAND PAGING ALLOCATION**

So far, the memory allocation schemes that we have learnt about have been requiring the entire program to be placed into memory. Demand paging allocation allows for only parts of the program to be placed into memory. Since it is a paging allocation scheme, all the jobs are first divided into pages. All these pages initially reside in secondary storage before being loaded into the main memory. During processing, the parts of the job or program that are needed are placed into memory.

A good example to explain demand paging is a situation where a user is working with Microsoft Word. When typing the document, a user may only work with one menu at a time while all the others stay idle. For instance, to change this WORD to bold, the user selects the home menu and selects B and gets **WORD**. All the other menu items are idle within Microsoft Word, therefore are not needed when converting a word to Bold. This is the premise that demand paging is built of. By allocating only portions of the job that are necessary for processing, demand paging ends up using less memory.

Page swapping refers to the movement of pages from main memory to secondary storage [1]. Operating systems rely on the JT, PMT and MMT to determine how pages are moved from one to the other. Figure 5 shows an example of Job H with its PMT and allocation in the main memory.

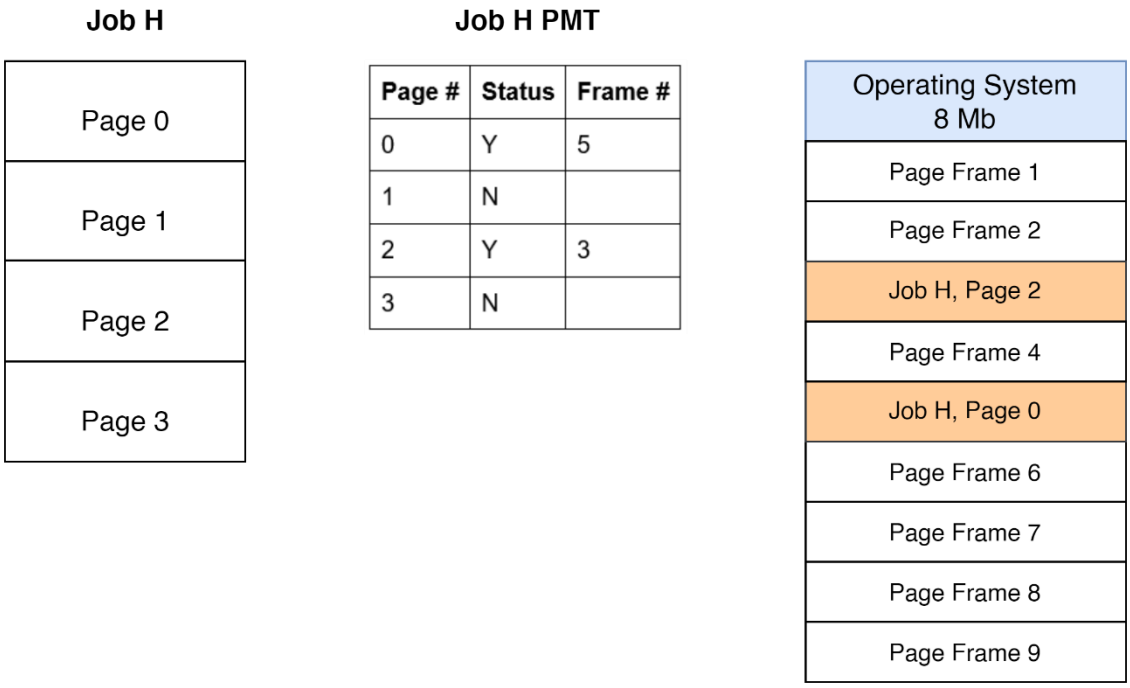


Figure 2: Demand Page allocation tables

Based on figure 2 above, we see that Job H has 4 pages. These pages are loaded onto the PMT table. The PMT lists whether the page is loaded in memory or not using the status field. In this example, pages 0 and 2 are loaded into frames 5 and 3 respectively, while pages 1 and 2 are not loaded.

**Advantage**

Demand paging does not require the entire program to be loaded into memory. This is an advantage as there is not a waste of memory and resources toward program pages occupying memory but remaining idle.

**Disadvantage**

One of the greatest disadvantages of demand paging is excessive page swapping. This occurs when pages are swapped too often between the main memory and the secondary memory. This is also known as **thrashing**. This leads to higher overheads by the operating system.

**PAGE REPLACEMENT POLICIES**

Paging requires the use of various policies that dictate how pages are swapped. Two main policies exist First In First Out (FIFO) and Least Recently Used (LRU).

FIFO works by replacing pages from frames that have been in memory longest or those that were in the earliest [1]. Assuming JOB H from Figure 2 needed to swap between two frames. We have pages 0 and Page 1 being loaded into Frame A and B. If page 2 needed to be swapped, it would be swapped with Page 0 as it was first placed into memory. Therefore Page 0 would be moved to the secondary storage and swapped with Page 2. With this method, it means now that Page 1 is the oldest page within memory and if Page 3 needed to be loaded it would be swapped with Page 1.

FIFO does not work efficiently especially when the page frames are few and there is excessive swapping. A page interrupt occurs when a page is loaded in memory and whether it is swapped out or not [1]. Increasing memory regardless of the policy is not a guarantee that swapping will reduce.

Least Recently Used (LRU): This allocation scheme is used to swap out pages that show the least activity and are less likely to be used in the future [1]. Let us review Job H again from the above example. There is a list that indicates when the page entered memory and was last used. In this example, any page frame will be checked for its activity and its ability to be required or used in the future. The number of swaps may reduce with a reduction in the number of pages interrupts as pages that are required more often are left within memory.

## SEGMENTED MEMORY ALLOCATION

Segmentation goes hand in hand with how programmers structure their programs into logical modules [1]. During segmented memory allocation, each job gets divided into segments of different sizes. These dynamic sizes are the key difference between the paging method and segmentation. Segmentation was developed to take care of page faults within the paging method [1].

With the jobs being divided into segments of various sizes, the memory is not divided, instead, segments are dynamically allocated the resources that they need in memory. This is like the dynamic partition discussed in an earlier lecture.

Like the paging memory allocation that used PMT, segmentation used a Segment Map Table (SMT). This table is used to store details about the segment including its numbers, length, status and when it is loaded into memory and its memory location [1]. Three tables, JT, SMT and MMT are used to support the operating system when keeping track of the segments.

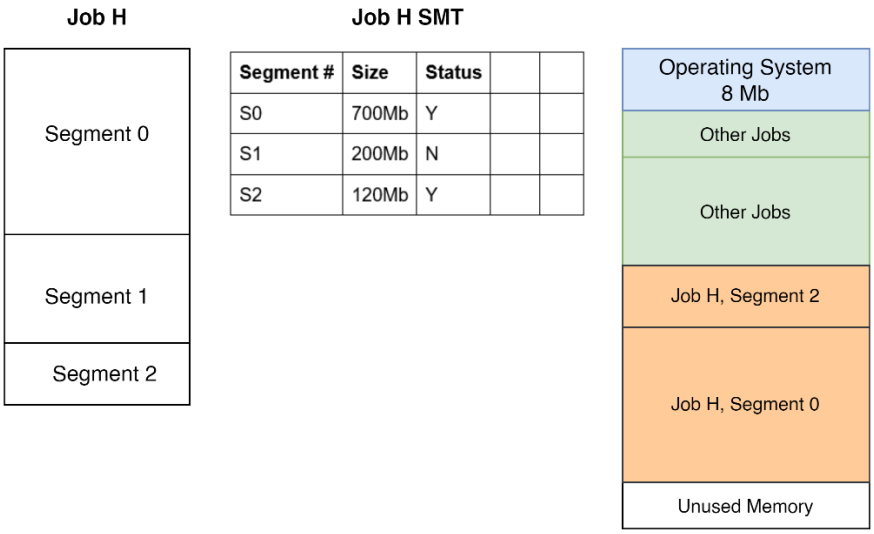


Figure 3: Segmented Memory Allocation

As shown in Figure 3, Job H is divided into segments of variable lengths. These segments are divided into variable sizes and then stored within the SMT. As with demand paging, only the needed segments are loaded into memory. Remember, the loading happens dynamically but in a non-contiguous way.

### Advantage

This method deals with the page faults from the paging method.

### Disadvantage

The method uses a dynamic method to allocate segments to memory. Dynamic allocation results in external fragmentation like dynamic partitioning. Compaction is needed to ensure that memory is not wasted.

### DIFFERENCE BETWEEN SEGMENTATION AND PAGING.

Both methods are similar. However, Paging uses pages and page frames in memory. Segmentation uses segments and memory is dynamically allocated.

Paging pages are physical units that are invisible to the user's program while segments are logical units that are visible to the user's program.

### SEGMENTED/DEMAND PAGED MEMORY ALLOCATION [1]

This memory allocation scheme evolved from the segmented memory and the demand paged memory. It picks the advantages of both and merges them into one. The same methods used in both are used here with some exceptions. The tables used are the JT, SMT, PMT and MMT. There is one JT for all the jobs in the system, each job has its own SMT and each SMT has its own PMT while the MMT is for the entire system. This is shown in Figure 4.

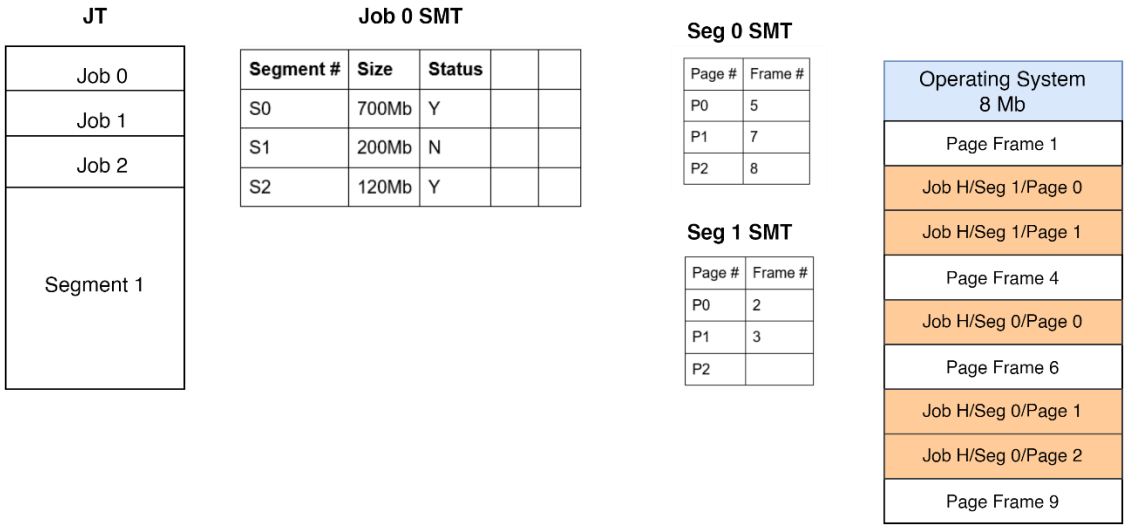


Figure 4: Segmented/Demand Paging Allocation

To access the location of a particular job, the scheme needs to access the Job number, then the segment number, and the page number for it to access the exact location.

The biggest drawback of this scheme is the need to access 4 tables meaning lots of overhead and the time it takes to reference the tables. One mitigation is the use of associative memory.

Associative memory is the name given to registers that are allocated to each active job [1]. The registers associate several segments and page numbers for a specific job and the main memory address [1]. Registers are generally hardware; therefore, the number of registers will be dependent on the type of machine.

## SUMMARY

This lecture reviewed memory management, and it featured the paging memory allocation scheme. Specifically, an overview of the scheme was done while reviewing the demand paging scheme. The two replacement schemes reviewed included the First In First Out (FIFO) and Least Recently Used (LRU). Finally, a review of segmentation was done with the segmentation/demand paging being reviewed last.

## DISCUSSION TOPIC

Throughout the lecture, we reviewed the various memory management schemes. Based on the computers or laptops that you are using, identify the various types of paging mechanisms being used. Remember this will be dependent on both the operating system and the hardware.

## REFERENCES

[1] McHoes, A., & Flynn, I., Understanding Operating Systems. Boston: Cengage Learning, 2018

[2] Stallings, W., Operating Systems: Internals and Design Principles. Harlow: Pearson Education Limited, 2018.