Part (4)

VIRTUAL MEMORY

***** *Virtual memory* : Is a technique for using the secondary storage(hard disk) to extend the apparent limited size of the physical memory(main memory).

***** If the segment of the program containing the word requested by the processor is not in the main memory at the time of the request, then such segment will have to be brought from the disk to the main memory.

*****The address issued by the processor in order to access a given word does not correspond to the physical memory space is called a **virtual (logical)** address.

*** (MMU)** is responsible for the translation of virtual addresses to their corresponding **physical addresses**.

*****Three address translation techniques can be identified. These are direct-mapping, associative- mapping, and set-associative-mapping.

Page:

*****Movement of data between the disk and the main memory takes the form of pages. A **page** is a collection of memory words, which can be moved from the disk to the MM when the processor requests accessing a word on that page.

***** A typical size of a page in modern computers ranges from 2K to 16K bytes.

*****A **page fault** occurs when the page containing the word required by the processor does not exist in the MM and has to be brought from the disk (like a cache miss).

Page table:

A page table It is a table which contains the mapping of virtual pages to physical pages and it is stored in the main memory. It contains:
* Modification of a page
* The authority for accessing a page.
* A bit indicating the validity of a page(**The valid bit).** It is set if the corresponding page is actually loaded into the main memory.
Valid bits for all pages are reset when the computer is first powered on.

* The other control bit that is kept in the page table is the **dirty bit**. It is set if the corresponding page has been altered while residing in the main memory. And reset If the page has not been altered.
This can help in deciding whether to write the contents of a page back into the disk (at the time of replacement) or just to override its contents with another page.

Translation Look-Aside Buffer (TLB) :

*In most modern computer systems a copy of a small portion of the page table is kept on the processor chip. This portion consists of the page table entries that correspond to the most recently accessed pages. This small portion is kept in **the translation look-aside buffer (TLB)** cache.

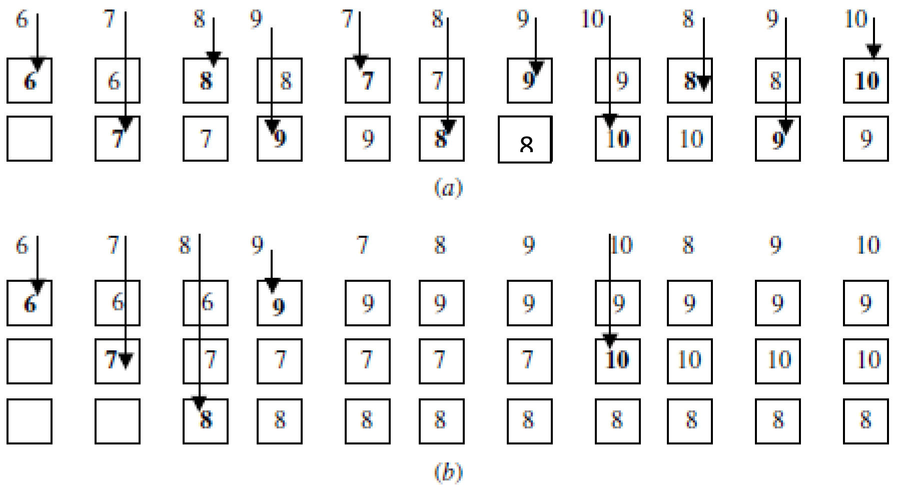
* A search in the TLB precedes that in the page table. Therefore, the virtual page field is first checked against the entries of the TLB in the hope that a match is found:
1-A hit in the TLB will result in the generation of the physical address of the word requested by the processor, thus saving the extra main memory access required to access the page table.
2-It should be noted that a miss on the TLB is not equivalent to a page fault.

*It is clear from the above discussion that as more requests for items that do not exist in the main memory (page faults) occur, more pages would have to be brought from the hard disk to the main memory. This will eventually lead to a totally filled main memory.

Replacement Algorithms (Policies) :

Basic to the implementation of virtual memory is the concept of demand paging. This means that the operating system, and not the programmer, controls the swapping of pages in and out of main memory as they are required by the active processes.
replacement policy: A technique used in the virtual memory that makes a decision When a process needs a nonresident page, the operating system must decide which resident page is to be replaced by the requested page.
To illustrate the use of the **FIFO** mechanism, we offer the following example

Example :Consider the following reference string of pages made by a processor:
6, 7, 8, 9, 7, 8, 9, 10, 8, 9, 10. In particular, consider two cases: (a) the number of page frames allocated in the main memory is **TWO** and (b) the number of page frames allocated are **THREE**.
The figure below illustrates a trace of the reference string for the two cases. As can be seen from the figure, when the number of page frames is TWO, there were **11 page faults** (these are shown in bold in the figure).
When the number of page frames is increased to THREE, the number of page

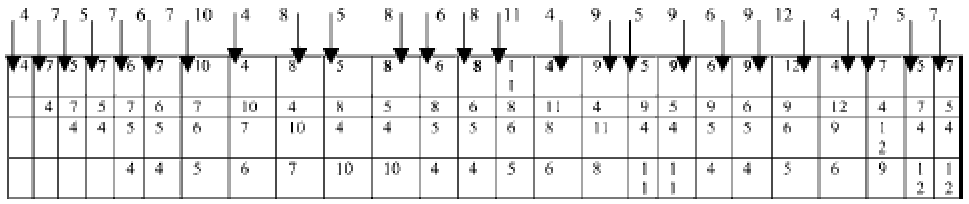


FIFO replacement technique. (a) FIFO replacement using two page frames (#PFs = 11), (b) FIFO replacement using three page frames (#PFs = 5)

faults was reduced to **five**. Since five pages are referenced, this is the optimum condition.

Least Recently Used (**LRU**) Replacement According to this technique, page replacement is based on the pattern of usage of a given page residing in the main memory <u>regardless of the time(spent)in the main memory</u>. The page that <u>has not been(referenced) for the longest time</u> while residing in the main memory is selected for replacement. To illustrate the use of the LRU mechanism, we offer the following example.

<u>Example:</u> Consider the following reference string of pages made by a processor:
4, 7, 5, 7, 6, 7, 10, 4, 8, 5, 8, 6, 8, 11, 4, 9, 5, 9, 6, 9, 12, 4, 7, 5, 7.
Assume that the number of page frames allocated in the main memory is **FOUR**. Compute the number of page faults generated. The trace of the main memory contents is shown in Figure below. Number of **page faults = 18.**
In presenting the **LRU**, we have a particular implementation, called stack-based LRU. In this implementation, the most <u>recently accessed page is now represented by</u>



4	7	5	7	6	7	10	4	8	5	8	6	8	11	4	9	5	9	6	9	12	4	7	5	7
4	7	5	7	6	7	10	4	8	5	8	6	8	11	4	9	5	9	6	9	12	4	7	5	7
	4	7	5	7	6	7	10	4	8	5	8	6	8	11	4	9	5	9	6	9	12	4	7	5
		4	4	5	5	6	7	10	4	4	5	5	6	8	11	4	4	5	5	6	9	12	4	4
				4	4	5	6	7	10	10	4	4	5	6	8	11	11	4	4	5	6	9	12	12

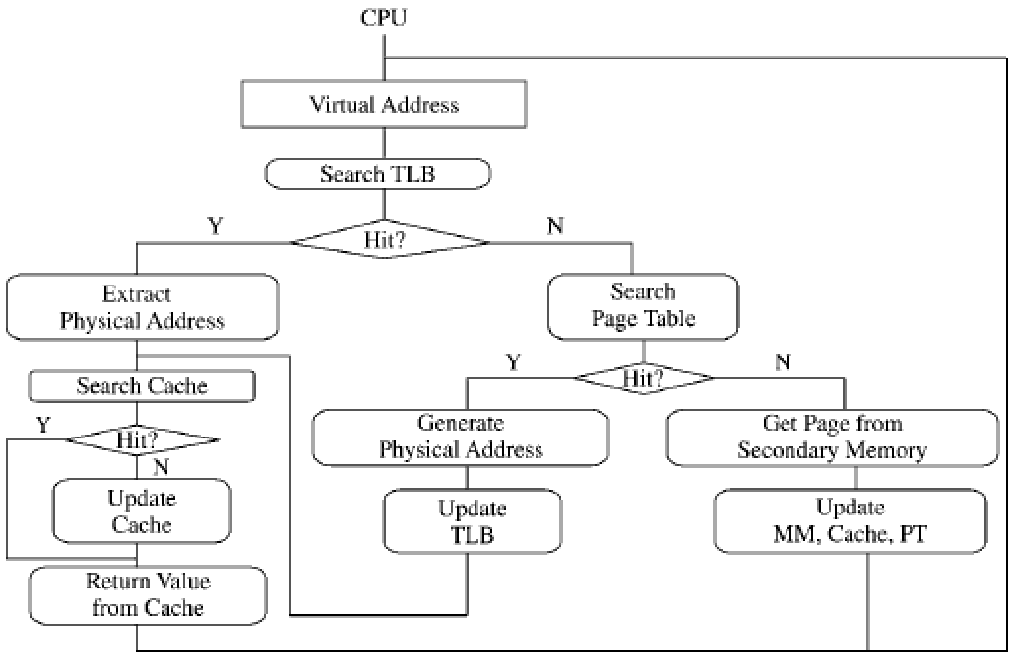
LRU replacement technique

<u>the top page rectangle</u>. The rectangles do not represent specific page frames as they did in the FIFO diagram. Thus, each reference generating a page fault is now on the top row.

<u>H.W:</u>

Virtual Memory Systems with Cache Memory :

A typical computer system will contain a cache, a virtual memory, and a TLB. When a virtual address is received from the processor, a number of different scenarios can occur, each dependent on the availability of the requested item in the cache, the main memory, or the secondary storage. The following figure shows a general flow diagram for the different scenarios.



Memory hierarchy accesses scenarios

Segmentation :

• A segment is a block of contiguous locations of varying size.

• Segments are used by the operating system (OS) to relocate complete programs in the main and the disk memory.

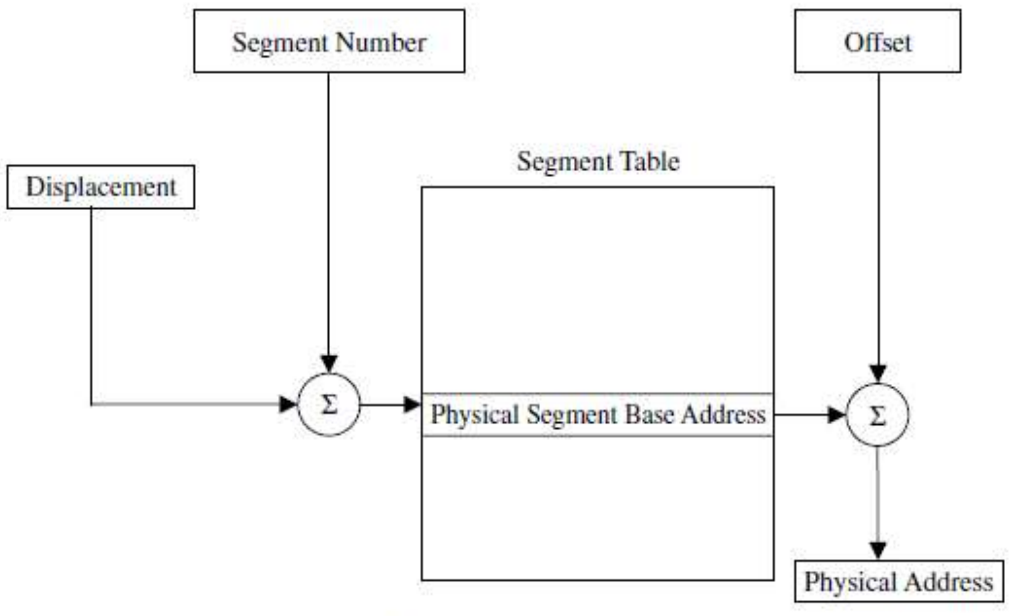
• Segments can be shared between programs.

• They provide means for protection from unauthorized access and/or execution. It is not possible to enter segments from other segments unless the access has been specifically allowed.

• Data segments and code segments are separated. It should also not be possible to alter information in the code segment while fetching an instruction nor should it be possible to execute data in a data segment.

Segment Address Translation:

In order to support segmentation, the address issued by the processor should consist of :a segment number and a displacement within the segment.



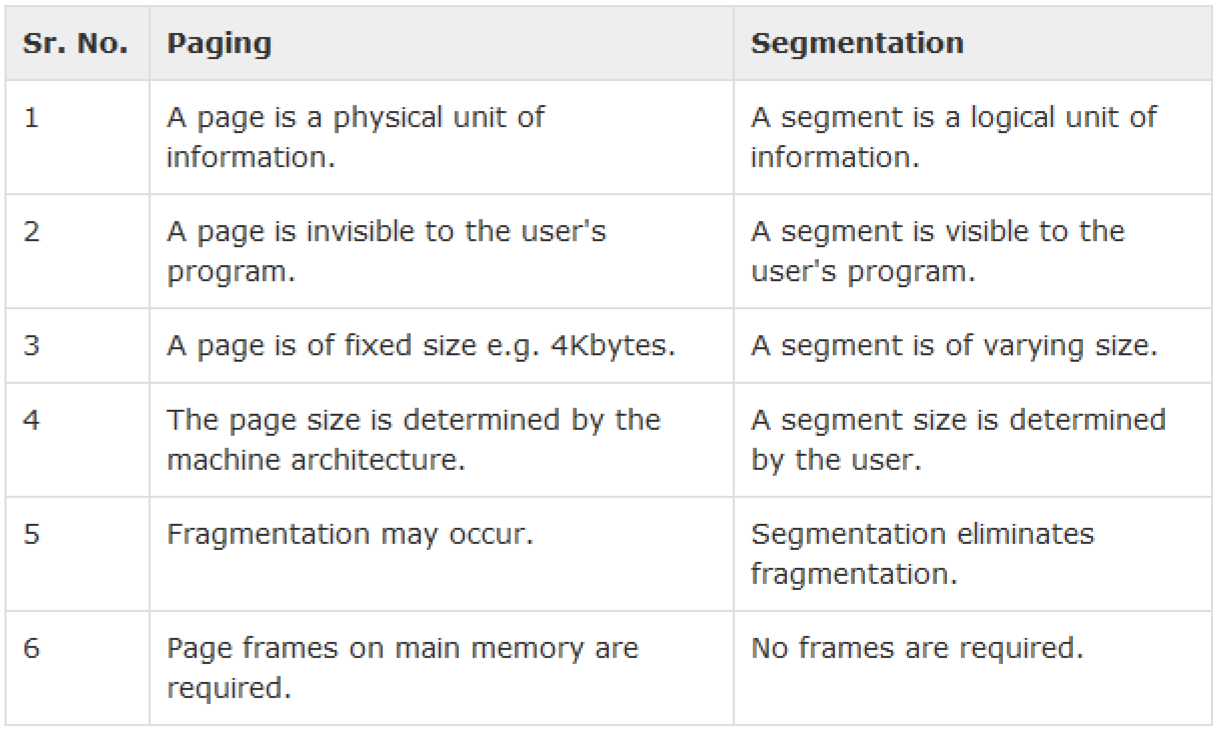
Segment address translation

Address translation is performed directly via a segment table.

***** The starting address of the targeted segment is obtained by:
adding the segment number to the contents of the displacement .

***** Adding the physical segment base address to the offset yields the required physical address

The differences between paging and segmentation.



Sr. No.	Paging	Segmentation
1	A page is a physical unit of information.	A segment is a logical unit of information.
2	A page is invisible to the user's program.	A segment is visible to the user's program.
3	A page is of fixed size e.g. 4Kbytes.	A segment is of varying size.
4	The page size is determined by the machine architecture.	A segment size is determined by the user.
5	Fragmentation may occur.	Segmentation eliminates fragmentation.
6	Page frames on main memory are required.	No frames are required.

Paged Segmentation:

• Both segmentation and paging are combined in most systems.

• Each segment is divided into a number of equal sized pages.

•The basic unit of transfer of data between the main memory and the disk is the page, that is, at any given time, the main memory may consist of pages from various segments.

In this case, the virtual address is divided into a segment number, a page number, and displacement within the page.

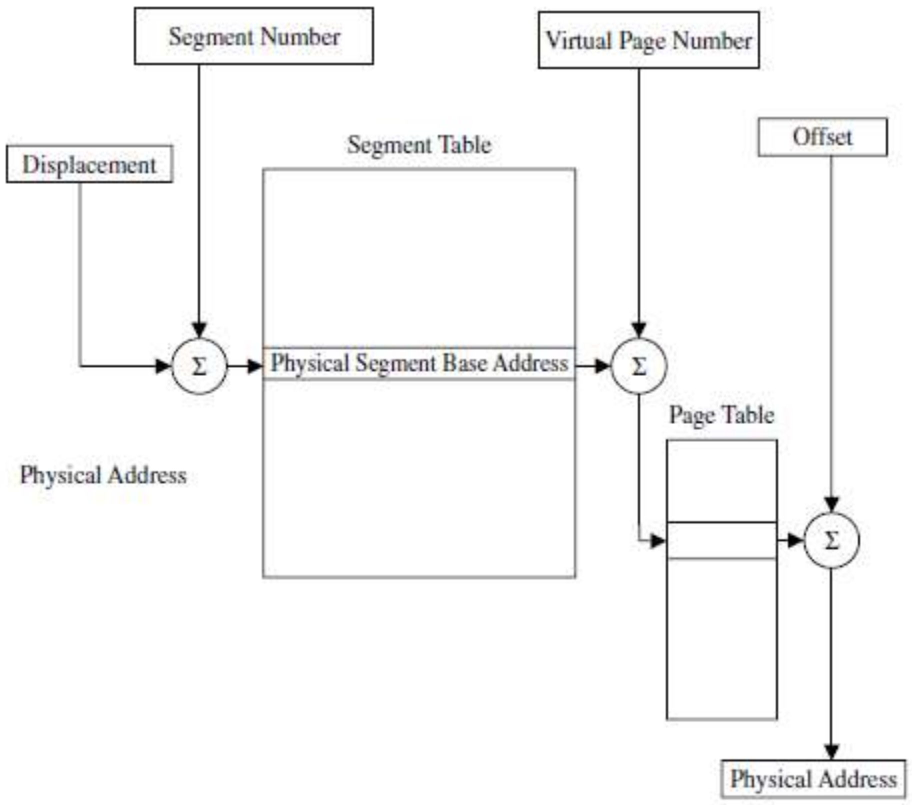


Figure 7.19 Paged segmentation address translation