Chapter 4

Memory Management and Virtual Memory

LEARNING OBJECTIVES

- *Basic concepts*
- *Memory management requirements*
- *Relocation and memory mapping techniques*
- *Placement algorithm*
- *Dynamic partitioning*
- *Placement algorithm*
- *Buddy system*
- *Non-contiguous storage allocation methods*
- *Paging*
- *Segmentation*
- *Page table structure*
- *Hierarchical page table*
- *Inverted page table*
- *Address translation in a segmentation system*

Basic Concepts

Uniprogramming system Main memory is divided into two parts as follows:

1. Operating system (OS) part
2. Program part (which is currently being executed)

Multiprogramming system Here the user part of memory must be further subdivided to accommodate multiple processes.

The task of subdivision is carried out dynamically by the OS and is known as memory management.

Memory Hierarchy

The triangle in Figure 1 gives the hierarchy of memory. The memory hierarchy shows the performance issues.

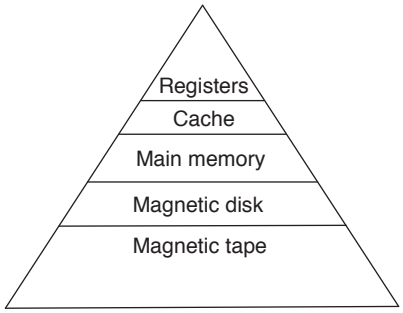


Figure 1 Memory hierarchy.

The memory hierarchy has different types of storage system in computers which are arranged in hierarchy, with respect to speed and cost.

If one moves down the hierarchy, access time increases, the cost per bit decreases, the memory capacity increases and memory access frequency by the processor decreases.

The registers, cache and main memory are volatile, whereas magnetic disc and magnetic tapes are non-volatile storage devices.

Memory Management Requirements

Memory management requirements are as follows:

1. Relocation
2. Protection
3. Sharing
4. Logical organization
5. Physical organization

Relocation

1. The role of relocation, the ability to execute processes independently from their physical location in memory, is central for memory management.
2. In a general purpose multiprogramming environment, a program cannot know in advance what processes will be running in memory when it is executed, nor how much memory the system has available for it, nor where it is located.
3. Hence program relocation is required such that a program must be compiled and linked in such a way that it can later be loaded starting from an unpredictable address in memory, an address that can even change during the execution of the process itself, if any swapping occurs.

4. The basic requirement for program relocation is that all the references to memory it makes during execution must not contain absolute (physical) address of memory cells, but must be generated relatively, that is, as a distance measured in number of contiguous memory words, from some known point.

Protection

Each process should be protected against unwanted interference by other processes, whether accidental or intentional.

Thus, programs in other processes should not be able to reference memory locations in a process for reading or writing purpose without permission.

Sharing

1. Any protection mechanism must have the flexibility to allow several processes to access the same portion of main memory.
2. Processes that are cooperating on some task may need to share access to the same data structure.
3. The memory management system must therefore allow controlled access to shared areas of memory without compromising essential protection.

Logical Organization

Main memory in a computer system is organized as a linear address space consisting of a sequence of bytes or words. But most of the programs are organized into modules.

If the OS and hardware can effectively deal with user programs and data in the form of modules, then there are some advantages.

1. Modules can be written and compiled independently.
2. Different degrees of protection can be given to different modules.
3. It is better to share modules among processes.

Physical Organization

Computer memory is organized in two levels:

1. Main memory
2. Secondary memory

The flow of information between these two modules is a major concern. If this is assigned to user, then there are some problems:

1. ***Overlaying*** may be possible. In overlaying concept, the various modules of a program can be assigned to the same region of memory, which causes wastage of programmer time.
2. The programmer does not know at the time of coding how much space will be available or where that space will be. So it must be handled by the system.

Address binding Addresses may be represented in different ways during the program execution:

1. Addresses in source program are generally symbolic.
2. A complier will typically bind these symbolic addresses to relocatable addresses.
3. The linkage editor or loader will in turn bind the relocatable addresses to absolute addresses.

So the binding of instructions and data to memory addresses can be done at any step along the way:

1. Compile time
2. Load time
3. Execution time

Compile time If you know at compile time where the process will reside in memory, then absolute code can be generated.

Load time If it is not known at compile time where the process will reside in memory, then the compiler must generate relocatable code.

Execution time If the process can be moved during its execution from one memory segment to another, then binding must be delayed until run time.

Logical versus Physical Address Apace

Logical address An address generated by the CPU is commonly referred to as a logical address.

Physical address An address seen by the memory unit, that is, one loaded into MAR is referred as a *physical address*.

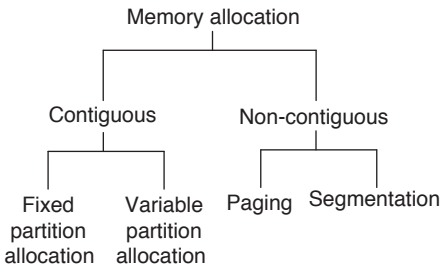
Notes:

1. Logical and physical addresses differ in execution time address-binding scheme.
2. Logical and physical addresses are same in compile time and load time address-binding schemes.
3. The run-time mapping from logical to physical address is done by a hardware device called the *memory management unit* (MMU).

Memory Mapping Techniques

The principle operation of memory management is to bring processes into main memory for execution by the processor. Let's now discuss various memory management techniques as follows:

1. Fixed partitioning
2. Dynamic partitioning
3. Simple paging
4. Simple segmentation
5. Virtual memory paging
6. Virtual memory segmentation.



Contiguous Storage Allocation

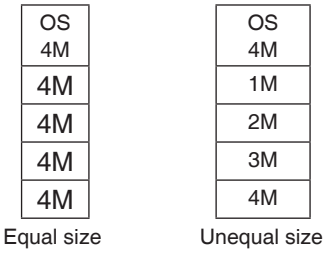
In this allocation, a memory resident program occupies a single contiguous block of memory.

Fixed/Static Partitioning

The main memory is divided into a number of static partitions at system generation time. Moreover, a process may be loaded into a partition of equal or greater size.

Partition size Two alternatives of fixed partition are as follows:

1. Equal-size partitions
2. Unequal-size partitions



Equal-size partitions: Any process whose size is less than or equal to the partition size can be loaded into any available partition.

Two problems with this technique are as follows:

1. A program may be too big to fit into a partition. Use overlaying to solve this problem.
2. Main memory utilization is extremely inefficient, as there is a possibility of internal fragmentation.

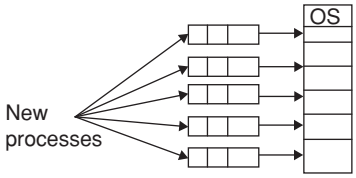
In *internal fragmentation*, there is a space wastage internal to a partition due to the fact that the block of data loaded is smaller than the partition.

Unequal-sized partition: Both the problems with equal-size partition can be lessened by using unequal-sized partitions.

Placement algorithm: With equal-size partitions, the placement of processes in memory is trivial. As all partitions are of equal size, it doesn't matter which partition is used.

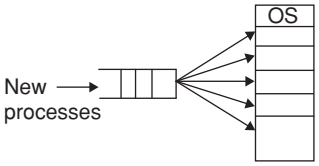
With Unequal-size partitions, there are two possible ways to assign processes to partitions:

1. Assign each process to the smallest partition within which it will fit.



- Figure shows one process queue for partition.
- Minimized internal fragmentation.
- Possibility of unused partitions.

2. Employ a single queue for all processes.



- When it is time to load a process into main memory, the smallest available partition that will hold the process is selected.

Advantages

1. Simple to implement.
2. Little OS overhead.

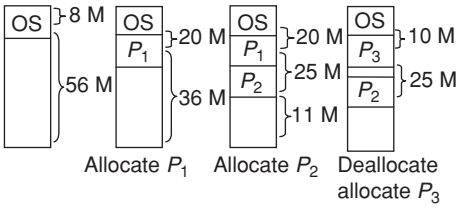
Disadvantages

1. Inefficient use of memory due to internal fragmentation.
2. Maximum number of active processes is fixed.

Dynamic Partitioning

With dynamic partitioning, the partitions are of variable length and number. When a process is brought into main memory, it is allocated exactly as much memory as it requires and no more.

Example:



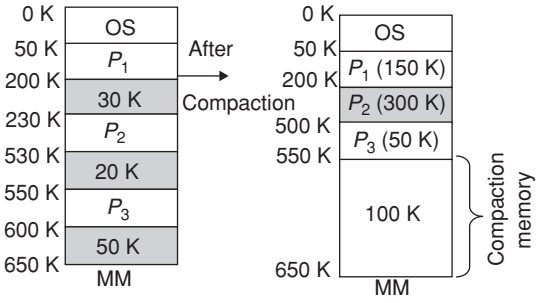
- This method starts out well, but eventually it leads to a situation in which there are a lot of small holes in memory.
- As time goes on, memory becomes more and more fragmented and memory utilization declines. This phenomenon is referred to as *external fragmentation*.

It indicates the memory that is external to all partitions becomes increasingly fragmented.

Compaction

Compaction is a technique by which the resident program are relocated in such a way that the small chunks of free

memory are made contiguous to each other and clubbed together into a single free partition that may be big enough to accommodate more programs.



It should be noted that compaction involves dynamic relocation of a program.

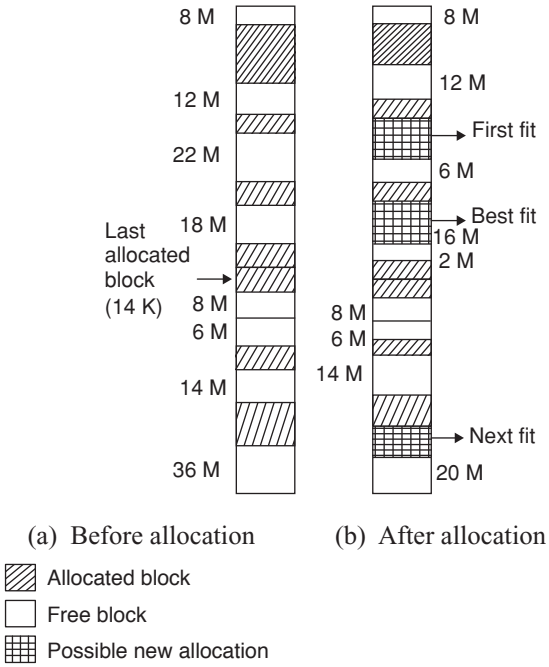
Placement algorithm

Memory compaction is a time-consuming process, and hence the OS uses some placement algorithms.

The three most common strategies to allocate free partitions to the new processes are as follows:

1. *First fit*: Allocate the first free partition, large enough to accommodate the process. IT executes faster.
2. *Best fit*: Allocate the smallest free partition that meets the requirement of the process. It achieves higher utilization of memory by searching smallest free partition.
3. *Worst fit*: Allocate the largest available partition to the newly entered process in the system.
4. *Next fit*: Start from current location in the list.

Example: Consider the following memory configuration after a number of placement and swapping out operations. The last block that was used was a 22 MB block from which a 14 MB partition was created. The figure (b) shows 16 MB allocation request.



Advantages of dynamic partitioning

1. Memory utilization is generally better as partitions are created dynamically.
2. No internal fragmentation as partitions are changed dynamically.
3. The process of merging adjacent holes to form a single larger hole is called coalescing.

Disadvantages

1. Lots of OS space, time, complex memory management algorithms are required.
2. Compaction time is very high.

Buddy system: Both fixed and dynamic partitioning schemes have drawbacks.

In Buddy system, memory blocks are available of size 2^K words, $L \le K \le U$, where,

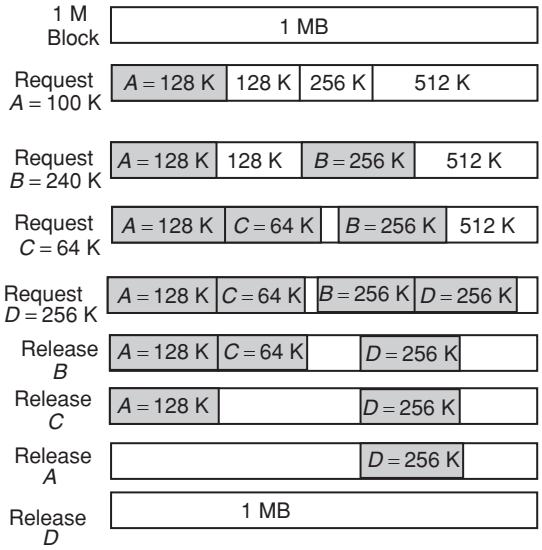
2^L = Smallest-size block that is allocated.

2^U = Largest-size block that is allocated.

Generally, 2^U is the size of the entire memory available for allocation. If a request of size'S' such that $2^{U-1} < S \le 2^U$ is made, then the entire block is allocated. Otherwise the block is split into two equal buddies of size 2^{U-1}. If $2^{U-2} < S \le 2^{U-1}$, then the request is allocated to one of the two buddies. Otherwise, one of the buddies is split in half again. This process continues until the smallest block greater than or equal to 'S' is generated and allocated to the request.

1. At any time, the buddy system maintains a list of holes of each size 2^i.
2. A hole may be removed from the $(i+1)$ list by splitting it in half to create two buddies of size 2^i in the 'i' list.
3. Whenever a pair of buddies on the i list both become unallocated, they are removed from the list and coalesced into a single block on the $(i+1)$ list.

Example:



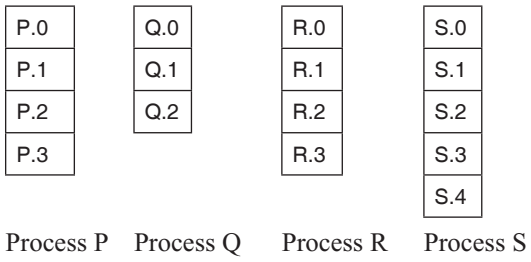
Non-contiguous Storage Allocation Methods

Paging

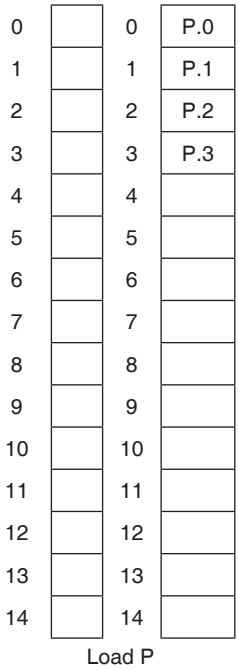
In simple paging, the main memory is divided into a number of equal-size frames. Each process is divided into a number of equal-size frames. The chunks of processes are referred as *pages*. A process is loaded by loading all of its pages into available, not necessarily contiguous frames.

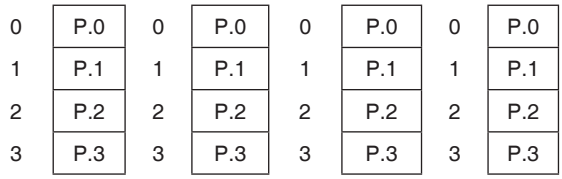
Example: At a point in time, some of the frames in memory are in use and some are free. A list of free frames is maintained by the OS.

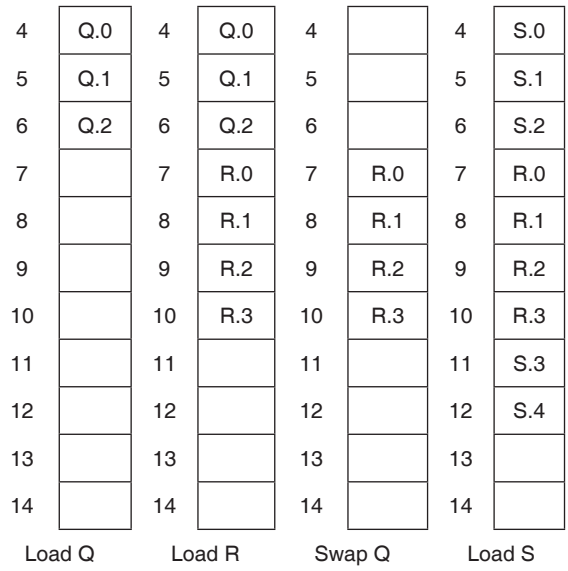
Consider four processes with their pages as displayed below:



Let the main memory consist of 15 frames:
Main memory



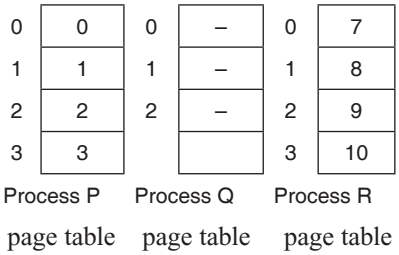


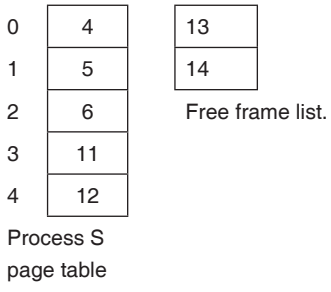


1. The OS maintains a page table for each process.
2. The page table shows the frame location for each page of the process.
3. Within a program, each logical address consists of a page number and an offset with in the page.
4. Here a logical address is the location of a word relative to the beginning of the program; the processor translates that into a physical address.
5. For this, the processor must know the following details:

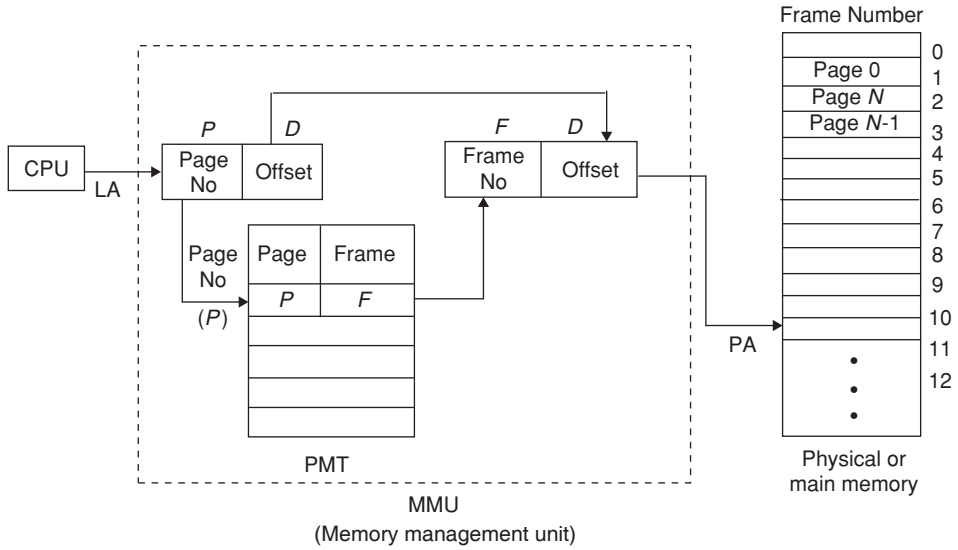
   - *Logical address:* Consists page number and offset.
   - *Page table:* Used to produce physical address (Frame number, offset).

In the previous example, the page tables of each process will be:



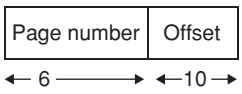


Address mapping in paging



Example: Let 16-bit address is used by the processor and the page size is 1 KB.

Then number of pages in main memory $= \frac{2^{16}}{2^{10}} = 2^6$.
∴ Page number = 6-bits
Offset = 10-bits



For the relative address 1502 =
0000010111011110
The page number = 000001 = 1
and offset = 0111011110 = 478, that is, the physical location will be an offset $(478)_{10}$ on page 1.

Let the page 1 is present in frame 6. Then the physical address will be 0001100111011110.

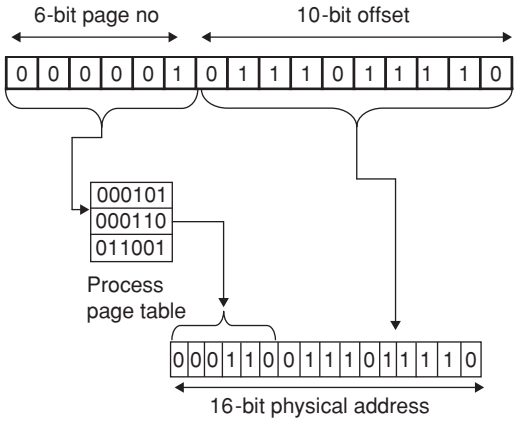


Figure 2 Paging.

Steps for address translation

1. Extract the page number from the logical address.
2. Use the page number as an index into the process page table to find the frame number.
3. The physical address will be constructed by appending the frame number to the offset.

Advantage

There is no external fragmentation.

Disadvantage

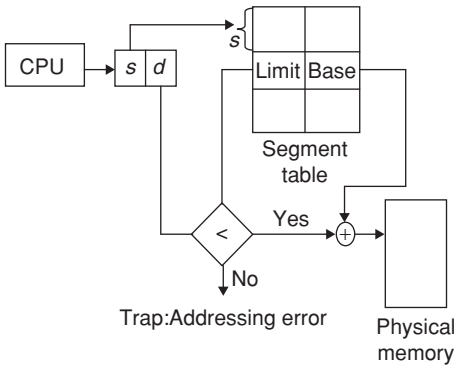
There is a small amount of internal fragmentation.

Segmentation

1. Each process is divided into a number of unequal-size segments.
2. A process is loaded by loading all of its segments into dynamic partitions that need not be contiguous.
3. The logical address using segmentation consists of two parts: *segment number* and an *offset*.
4. The principle inconvenience of segmentation is that the programmer must be aware of the maximum segment size limitation.
5. It makes use of a segment table for each process and a list of free blocks of main memory.
6. Each segment table entry would have to give the starting address in main memory of the corresponding segment. It also contains the length of the segment.
7. Steps for address translation:
   - Extract the segment number from the logical address.
   - Use the segment number as an index into the process segment table to find the starting physical address of the segment.

- Compare the offset to the length of the segment. If the offset is greater than or equal to the length, the address is invalid.
- The desired physical address is the sum of the starting physical address of the segment plus the offset.

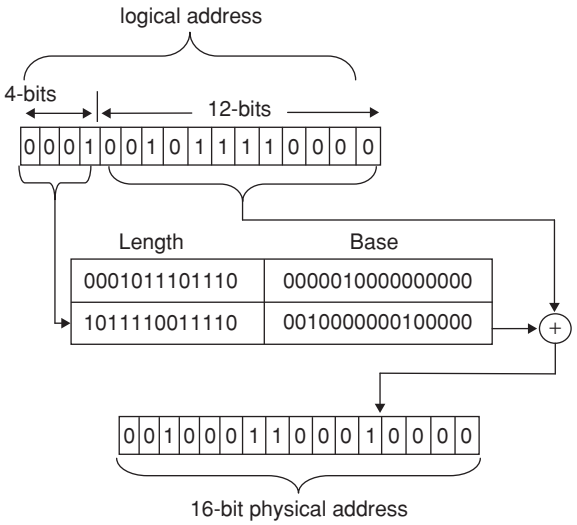
Hardware support for segmentation



Example: Consider the logical address 0001001011110000.

Let the segment number consists of 4-bits. Then segment number = 0001 = 1

Offset = 001011110000 = 752.



Advantages

1. No internal fragmentation
2. Improved memory utilization.
3. Reduced overhead compared to dynamic partitioning.

Disadvantage

1. External fragmentation.

Virtual Memory

1. In simple paging/segmentation, it is not necessary that all of the pages or all of the segments of a process be in main memory during execution.
2. Suppose that it is time to bring new process into memory. The OS begins by bringing in only one or a few pages to include the initial program page and initial data page to which those instructions refer.
3. The portion of a process that is actually in main memory at any time is defined to be the resident set of the process.
4. If the processor encounters a logical address that is not in main memory, it generates an interrupt indicating a memory access fault.
5. Then the OS brings the required page to the main memory.
6. With virtual memory,
   - More processes may be maintained in main memory.
   - A process may be larger than all of main memory, then also it will be executed.
7. Virtual memory is a storage allocation scheme in which secondary memory can be addressed as though it were part of main memory.
8. ***Thrashing:*** When the OS brings one page in, it must throw another out. If it throws out a page just before it is used, then it will just have to go get that piece again almost immediately. Too much of this leads to a condition known as *thrashing*.

   If the system spends most of its time in swapping rather than executing instructions then that situation refers to thrashing.
9. Principle of locality suggests that a virtual memory scheme may work.
10. Virtual memory will be practical and effective if
    - There is a hardware support for paging/segmentation.
    - The OS includes software for managing the movement of pages/segments.

Paging with Virtual Memory

1. The main difference between paging and virtual memory paging is that in virtual memory paging concept, not all pages of a process need to be in main memory frames for the process to run. Pages may be read in as needed.
2. A page table is also needed for a virtual memory scheme based on paging. Also it is typical to associate a unique page table with each process.
3. The virtual address and page table entries for virtual memory paging are shown below:

Virtual address

Page number	Offset

Page table entry

P	*M*	Other control bits	Frame number

P: Present bit. This bit specifies whether that particular page is present in main memory or not.

M: Modified bit. This bit indicates whether the contents of the corresponding page have been altered since the page was last loaded into main memory.

Page table structure

1. To read a word from memory, translate the virtual or logical address consisting of page number and offset into physical address consisting of frame number and offset, using a page table.
2. Page table must be stored in main memory to access it.

Hardware implementation for virtual memory paging

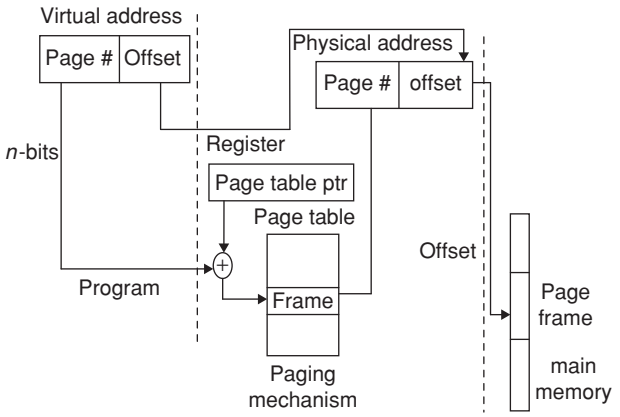
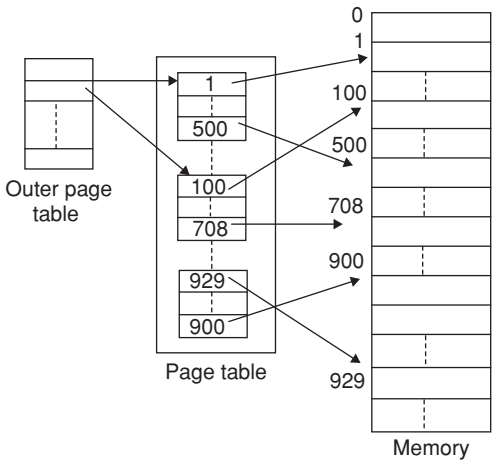


Figure 3 Address translation in paging system.

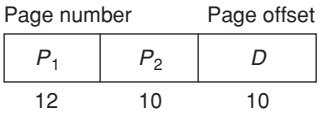
1. The amount of memory used by the page tables could be high.
2. To overcome this problem, most virtual memory schemes store page tables in virtual memory rather than real memory, that is, page tables are subject to paging just as other pages are.
3. When a process is running, at least a part of its page table must be in main memory, including the page table entry of the currently executing page.
4. Some processors make use of a two-level scheme to organize large page tables.

Hierarchical page table

1. If page table size is large, then use hierarchical page table.
2. The logical address space is broken up into multiple page tables.

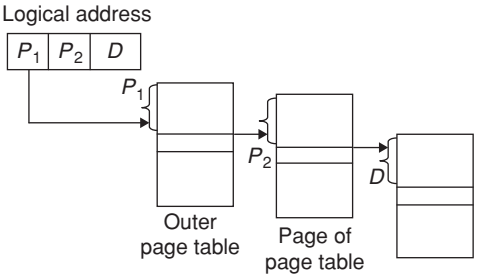


A logical address space (on 32-bit machine with 1 K page size) is divided into a page number consisting of 22-bits, page offset consisting of 10-bits. The page number is paged, the page number is divided into 12-bit page number and 10-bit page offset.



Here, P_1 is an index into the outer page table, P_2 is the displacement within the page of the outer page table.

Address translation (diagrammatic)



Drawback: Page table size is proportional to that of the virtual address space.

Inverted page table

1. Here, the page number portion of a virtual address is mapped into a hash value, using simple hash function.
2. The hash value is a pointer to the inverted page table, which contains the page table entries.
3. There is one entry in the inverted page table for each real memory page frame rather than one per virtual page.
4. Thus, a fixed proportion of real memory is required for the tables.
5. One virtual address may map into the same hash table entry, so a chaining technique is used for managing the overflow.
6. The page table's structure is called *inverted*, because it indexes page table entries by frame number rather than by virtual page number.

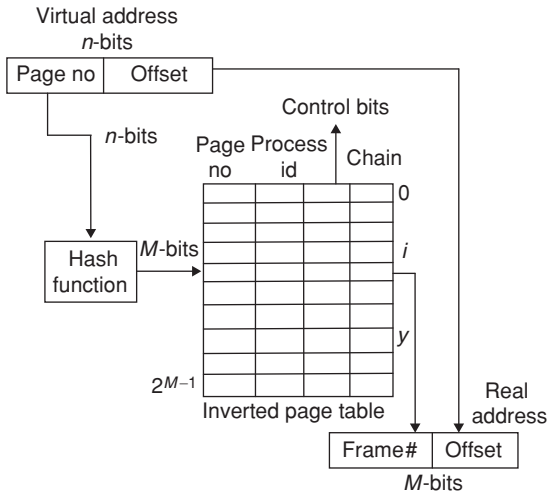


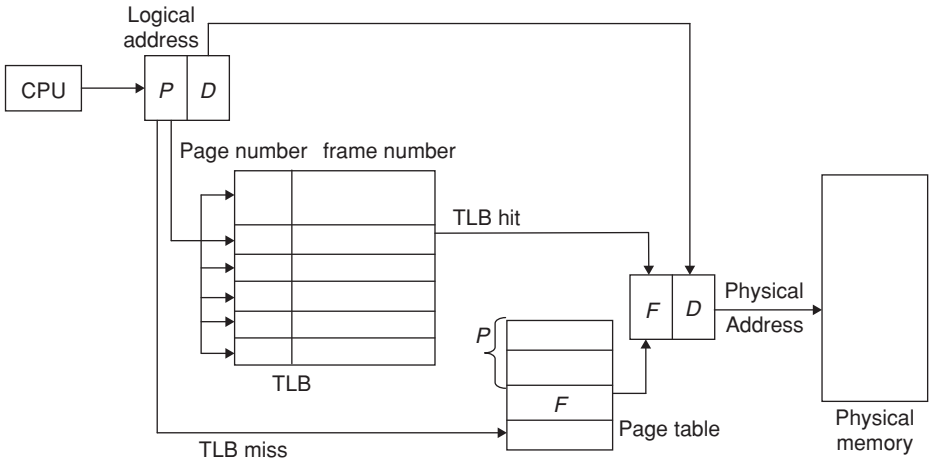
Figure 4 Inverted page table structure.

Translation look-a-side buffer (TLB)

1. The straight-forward virtual memory scheme would have the effect of doubling the memory access time.
2. To overcome this problem, most virtual memory schemes make use of a special high speed cache for page table entries, usually called TLB.
3. TLB contains the page table entries that have been most recently used.

Paging hardware with TLB

1. Given a virtual address, the processor will first examine the TLB. If the desired page table entry is present, that is, TLB hit, then the frame number is retrieved and real address is formed.
2. If there is a TLB miss, the processor uses the page number to index the process page table and examine the corresponding page table entry.
3. If present bit is set, then the page is in main memory and the processor can retrieve the frame number from the page table entry to form the real address.
4. The processor also updates the TLB to include this new page table entry.
5. If the page is not in main memory, then page fault is issued.
6. Then the OS will load the needed page and updates the page table.



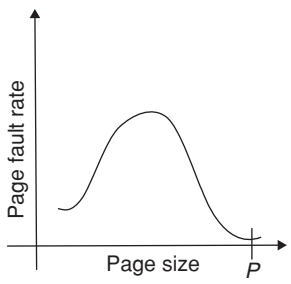
Organization of TLB

1. Each entry in TLB must include the page number and the complete page table entry.
2. The TLB may organized its entries either in
   - Direct mapping
   - Associative Mapping
3. The hardware must also consider the ways in which entries are organized and which entry to replace.

Note: The virtual memory mechanism must interact with the main memory cache system also.

Page size The factors to be considered for page size are as follows:

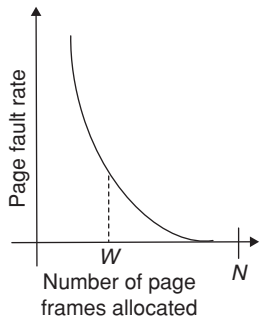
1. If page size is smaller: Then internal fragmentation is less. But it results in larger page tables. For large programs, the page fault rate increases.
2. Rotational secondary devices favour a larger page size for more efficient block transfer.



where P = size of entire process.

The above figure shows the relationship between page size and page fault rate.

The page fault rate is also determined by the number of frames allocated to a process. This relation is shown below.



where W = working set size
N = Total number of pages in process

Note: The design issue of page size is related to the size of physical main memory and program size.

Advantages of virtual memory paging

1. No external fragmentation
2. Higher degree of multiprogramming
3. Large virtual address space

Disadvantage

Overhead of complex memory management.

Segmentation with Virtual Memory

1. Memory consists of multiple segments.
2. Segments are of unequal and dynamic in size.
3. It simplifies the handling of growing data structures.
4. It allows programs to be altered and recompiled independently.
5. It lends itself to sharing among processes.
6. It lends itself to protection.
7. A unique segment table is associated with each process.
8. The virtual address and segment table entries are as shown below:

Virtual address

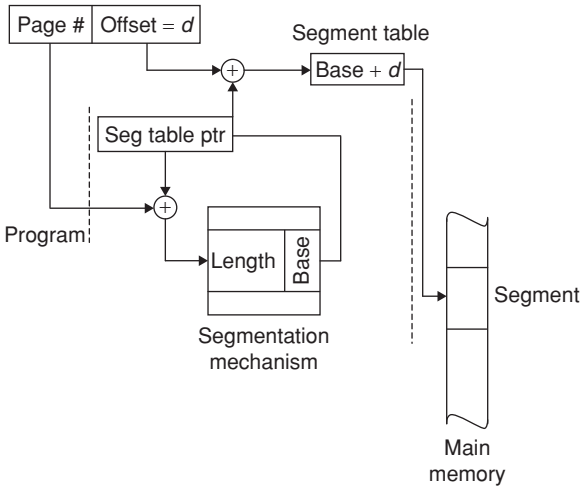
Segment number	Offset

Segment table entry

P	M	Other control bits	Length	Segment base

9. Only some of the segments of a process may be in main memory. To identify which segment is present in the main memory, use present bit P.
10. To know whether the segment is modified or not, use M-bit.

Address translation in a segmentation system (using virtual memory)



Advantages

1. No internal fragmentation
2. Higher degree of multiprogramming
3. Large virtual address space.
4. Protection and sharing support.

Disadvantage

1. Overhead of complex memory management.

Combined paging and segmentation

1. Here, the users address space is broken up into a number of segments by the programmer.
2. Each segment is, in turn, broken up into a number of fixed size pages, which are equal in length to a main memory frame.
3. If a segment has length less than that of a page, the segment occupies just one page.
4. The virtual address, segment table and page table entries are as shown below:

Virtual address

Segment number	Page number	Offset

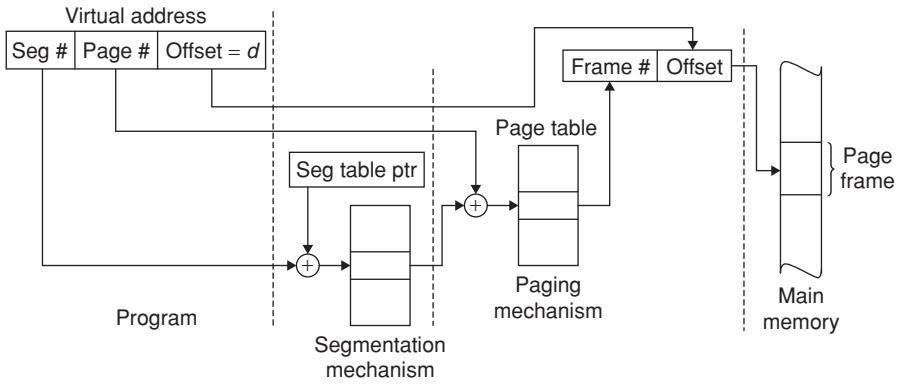
Segment table entry

Control bits	Length	Segment base

Page table entry

P	M	Other control bits	Frame number

Structure of combined segmentation/paging system



Notes:

1. Segmentation lends itself to the implementation of protection and sharing policies.
2. To achieve sharing, it is possible for a segment to be referenced in the segment tables of more than one process.

OS Software for Memory Management

We consider the following software policies for virtual memory:

1. Fetch policy:
   - Demand
   - Prepaging
2. Placement policy
3. Replacement policy:
   - Optimal
   - LRU
   - FIFO
   - Clock
4. Resident set management
   - Resident set size
   I. Fixed
   II. Variable
   - Replacement scope
   I. Local
   II. Global
5. Cleaning policy
   - Demand
   - Pre-cleaning
6. Load control
   - Degree of multi-programming

Fetch policy Determines when a page should be brought into main memory.

1. ***Demand paging:*** Here a page is brought into main memory only when a reference is made to a location on that page.
2. ***Prepaging:*** It is a technique that reduces the large number of page faults at process start up.
   - Prepaging is used to get before all or some of the pages a process will need, before they are referenced.
   - If prepaged pages are unused, I/O and memory would be wasted.
   - Assume 's' pages are prepaged and α of the pages are used.
   - Cost of ($s * \alpha$) to save page faults greater or less than the cost of prepaging $s \times (1 - \alpha)$ unnecessary page. If α near zero $\Rightarrow$ prepaging is lost.

Placement Policy

1. Determines where in real memory a process piece is to reside.
2. In pure segmentation system, the policies like best fit, first fit, etc., are used.
3. For a system that uses either pure paging or paging combined with segmentation, placement is usually irrelevant.

Replacement policy This deals with the selection of a page in main memory to be replaced when a new page must be brought in. In a replacement policy, we have to consider the following:

1. How many page frames are to be allocated to each active process.
2. Whether the set of pages to be considered for replacement should be limited to those of the process that caused the page fault or encompass all the page frames in main memory.
3. Among the set of pages considered, which particular page should be selected for replacement.

Page Fault

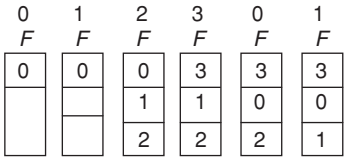
1. Whenever a processor needs to execute a particular page and that page is not available in main memory, this situation is said to be *page fault*.

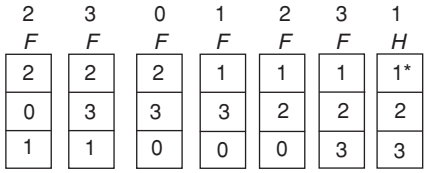
2. When the page fault occurs, the page replacement will be done.
3. 'Page Replacement' means select a victim page in the main memory, replace that page with the required page from the backing store (disk).
4. Some of the replacement algorithms are as follows:
   - FIFO
   - Optimal
   - LRU
   - Clock

FIFO (First-in-First-Out Algorithm)

1. Replace a page that is the oldest page of all the pages of the main memory.
2. Focuses on the length of time a page has been in memory rather than how much the page is being used.

Example:
Consider the reference string: 0 1 2 3 0 1 2 3 0 1 2 3 1





Here the symbol '*F*' indicates page fault.

The number of page faults = 12

'*H*' indicates the page is already in the memory. The remaining pages are not present in memory that is why page fault occurs. In general, the more frames there are, the less page fault.

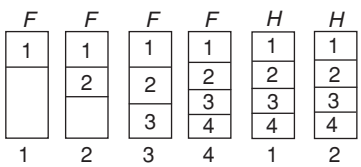
$$\text{Page fault rate} = \frac{\text{Number of page faults}}{\text{Number of bits in reference string}}$$

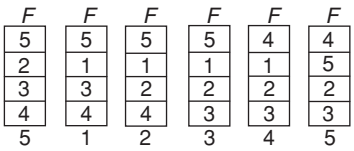
$$= \frac{12}{13} = 0.923 = 92.3\%.$$

Belady's Anomaly

Example: Consider the reference string: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

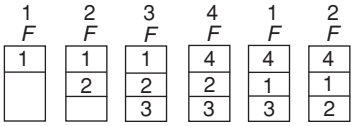
Number of frames = 4

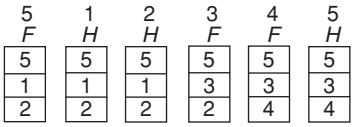




The number of page faults = 10

Consider the same reference string with three frames.





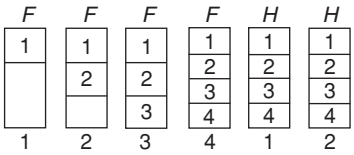
Here number of page faults = 9

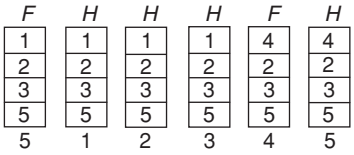
Here, as the number of frames increases the page fault also increases. This is known as *Belady's anomaly*.

Optimal page replacement algorithm

1. Replace the page that will not be used for the longest period of time.

   Example: Consider the reference string: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5 using four frames.





Number of page faults = 6

Disadvantage

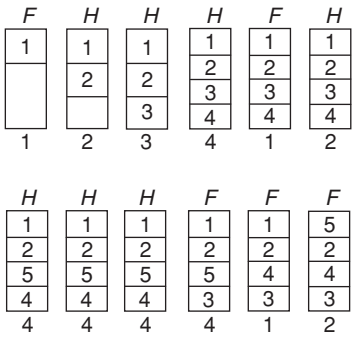
It requires future knowledge of reference string, so used for comparison studies.

LRU (least recently used) algorithm

1. Replace a page that has not been used for the longest period of time.
2. It looks backward in time rather than forward.
3. It associates with each page the time of that page last used.
4. Two methods of implementation:
   - Counters
   - Stack

   Example: Consider reference String: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

Number of frames = 4



Number of page faults = 8 (less than FIFO)

Approach to Implement LRU Replacement

LRU is a good page replacement policy, but the problem with these is how to determine the frame used for the last time.

This is implemented by using two approaches:

1. Using counters
2. Using stacks

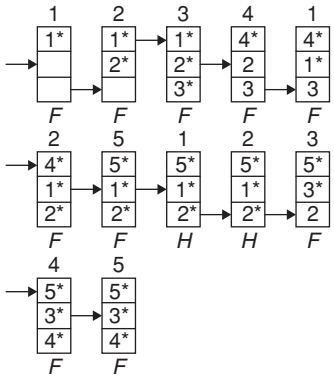
Using counters, LRU is implemented as follows: Every page entry has a counter, whenever a page is referenced, the clock value is copied into the counter. If the page has to be replaced, then it refers to the look up of the counter, whichever is having oldest time that is changed.

Using stack, LRU is implemented as follows: Form a doubly linked list of page numbers and keep it in stack whenever a page is referenced it is moved to the top of the stack that is top of the stack contains recently referenced page. Bottom of the stack will have least recently used one.

Clock replacement algorithm

1. The simplest form of clock policy requires the association of an additional bit with each frame, referred to as the use bit.
2. When a page is first loaded into frame in memory, the use bit for that frame is set to 1.
3. Whenever the page is subsequently referenced, its use bit is set to 1.
4. The set of frames that are candidates for replacement is considered to be a circular buffer, with which a pointer is associated.
5. When a page is replaced, the pointer is set to indicate the next frame in the buffer after the one just updated.
6. When it comes time to replace a page, the OS scans the buffer to find a frame with a use bit set to 0.
7. Each time it encounters a frame with a use bit of 1, it resets that bit to 0 and continues on.
8. If any of the frames in the buffer have a use bit of 0 at the beginning of this process, the first such frame encountered is chosen for replacement.
9. If all of the frames have a use bit of 1, then the pointer will make one complete cycle through the buffer, setting all the use bits to 0 and stop at its original position, replacing the page in that frame.

Example: Consider the reference string 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5, with three main memory frames:



∴ Number of misses = 9

Note: The clock algorithm was approximately closer in performance to LRU.

Effective memory access time

The percentage of times a page number is found in the associative registers is called the *hit ratio*. If we fail to find the page number in the associative registers, then we must first access memory for the page table and frame number, and then access the required byte in memory. To find the effective access time, we should weigh each case by its probability.

EMAT: Is given as = $p * s + (1 - p) * m$.
Where
p = Page fault rate
s = Page fault service time
m = Main memory access time
$(1 - p)$ = page hit ratio.

Frame locking Some of the frames in main memory may be locked. When a frame is locked, the page currently stored in that frame may not be replaced.

Page buffering To improve performance, a replaced page is not lost but rather is assigned to one of two lists:

1. The free page list if the page has not been modified or
2. The modified page list if it has modified.

Resident set management

Resident set size The OS must decide how many pages to bring in, that is, how much main memory to allocate to a particular process. Two policies are there:

1. Fixed allocation
2. Variable allocation

Fixed allocation This policy gives to a process a fixed number of frames in main memory within which to execute.

Variable allocation This policy allows the number of page frames allocated to a process to be varied over the life time of the process.

Replacement scope This is of two types as follows:

Local page replacement: When a process requests for a new page to be brought in and there are no free frames in the memory, we choose a frame allocated to only that process for replacement.

Global replacement: It allows a process to select a replacement frame from the set of all frames, even if that frame is currently allocated to some other processes. So, one process can take a frame from another.

Fixed allocation, local replacement

1. Number of frames allocated to a process is fixed.
2. Page to be replaced is chosen from among the frames allocated to that process.

Variable allocation, global scope

1. Page to be replaced is chosen from all available frames in main memory.
2. Size of resident set of processes varies.

Variable allocation, local scope

1. The number of frames allocated to a process may be changed from time to time.
2. Page to be replaced is chosen from among the frames allocated to that process.

Working set: This strategy is used to determine the resident set size and the timing of changes.

The working set with parameter Δ for a process at virtual time t, which we designated as $W(t, \Delta)$, is the set of pages of that process that have been referenced in the last Δ virtual time units.

Virtual time: Consider a sequence of memory references, $r(1)$, $r(2)$, … in which $r(i)$ is the page that contains the ith virtual address generated by a given process.

Time is measured in memory references; thus, $t = 1, 2, 3, \ldots$ measures the processes internal virtual time.

The variable 'Δ' is a window of virtual time over which the process is observed.

The working set size will be a non-decreasing function of the window size.

$$W(t, \Delta + 1) \supseteq W(t, \Delta).$$

For the sequence of page references 24, 15, 18, 23, 17, 15, 24, 18, 17, 17, 15. And window size = 2 then working set will be

{24, {24, 15,}, {15, 18}, {18, 23}, {23, 24}, {24, 17}, {17, 18}, {18, 24}, {18, 24}, {18, 17}, {17}, {17, 15}}

Page fault frequency (PFF) algorithm

1. The algorithm requires a use bit to be associated with each page in memory.
2. The bit is set to 1, when that page is accessed.
3. When a page fault occurs, the OS notes the virtual time since the last page fault for the process.
4. A threshold F is defined. If the amount of time since the last page fault is less than F, then a page added to resident set of the process.
5. Otherwise, discard all pages with a use bit of 0 and shrink the resident set according.

Note: PFF does not perform well during the transient periods when there is a shift to a new locality.

Variable interval sampled working set (VSWS)

1. The VSWS policy evaluates the working set of a process at sampling instances based on elapsed virtual time.
2. VSWS considers three parameters:
   M: The minimum duration of sampling interval
   L: The maximum duration of sampling interval
   Q: The number of page faults that are allowed to occur between sampling instances.
3. The policy works as following:
   - If virtual time since the last sampling instance reaches L, then suspend the process and scan the use bits.
   - If, prior to an elapsed virtual time of L, Q page faults occur,
     I. If the virtual time since the last sampling instance is less than M, then wait until the elapsed virtual time reaches M to suspend the process and scan the use bits.
     II. If the virtual time since the last sampling instance is greater than or equal to M, suspend the process and scan the use bits.

Cleaning policy It determines when a modified page should be written out to secondary memory. Two approaches are as follows:

1. Demand cleaning
2. Pre-cleaning

Demand cleaning A page is written out to secondary memory only when it has been selected for replacement.

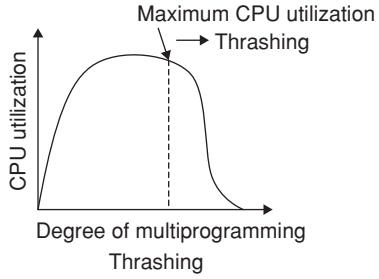
Pre-cleaning This policy writes modified pages before their page frames are needed so that pages can be written out in batches.

Load control It is concerned with determininig the number of processes that will be resident in main memory, which has been referred to as the multiprogramming level.

If too few processes are resident at any one time, it leads to swapping. If too many processes are present, thrashing will occur.

Cause of thrashing Consider the following scenario:

The OS monitors CPU utilization. If the utilization is too low, we increase the degree of multiprogramming by introducing a new process to the system. A global page replacement algorithm is used, which replaces pages with no regard to the process to which they belong. Now, say that a process enters a new phase in its execution and needs more frames. It starts faulting and taking frames away from other processes. These processes need those pages, however and so they also fault, taking frames from other processes. These faulting processes must use the paging device to swap pages in and out. As they queue up for paging device, the ready queue empties. As processes wait, for the paging device, CPU utilization decreases the CPU scheduler sees the decreasing CPU utilization; so it increases the degree of multiprogramming. The new process tries to get started by taking frames from running processes, causing more page faults, and a longer queue for paging device. As a result, CPU utilization drops even, further. The CPU scheduler tries to increase the degree of multiprogramming even more. Thrashing occurs, and the system throughput plunges. The page fault rate (FT) increases tremendously. Effective memory access time increases. No work is getting done because the processes are spending all their time in paging.



EXERCISES

Practice Problems I

Directions for questions 1 to 20: Select the correct alternative from the given choices.

1. Consider a 3-level memory hierarchy shown in the following table, with access times to memory:

Hierarchy level	Cache hit ratio	Page transfer time
M_1	0.55	0.003 ms
M_2	0.92	0.3 ms
M_3	–	1.0 ms

When a miss occurs, data is fetched from the next level. Calculate the average time required for a process to read one word from the memory system.

(A) 0.379 (B) 0.162
(C) 0.2798 (D) 0.172

2. Consider a memory system with FIFO page replacement algorithm policy. For an arbitrary page access pattern, increasing the number of page frames in main memory will

(A) Always decrease the number of page faults
(B) Always increase the number of page faults
(C) Sometimes increase the number of page faults
(D) Never effect the number of page faults

3. Consider the below page address stream generated by executing a program:

4 5 4 3 7 4

Assuming that LRU is used for page replacement and at most three frames are available in the memory for the process, find the number of page faults that can occur (initially all frames empty).

(A) 0 (B) 1
(C) 3 (D) 4

4. Consider a logical address space of four pages of 2048 words each mapped into a physical memory of 32 frames. How many bits in logical address?

(A) 12-bits (B) 14-bits
(C) 13-bits (D) 11-bits

5. The time taken to service a page fault is on average 10 ms and the memory access time is 20 µs. If the hit ratio is 70%, calculate the average access time.

(A) 3018 µs (B) 4014 µs
(C) 3014 µs (D) 4024 µs

6. Consider the following page trace:

4, 3, 2, 1, 4, 3, 5, 4, 3, 2, 1, 5

Number of frames for the Job $M = 4$. Then the page fault ratio using FIFO technique will be

(A) 63% (B) 75%
(C) 83% (D) 94%

7. The available main memory for loading pages is 64 MB with a frame size of 8 MB. If pages of size 6 MB, and 4 MB are loaded into memory, what is the percentage of the internal fragmentation resulted?

(A) 42.5 (B) 37.5
(C) 57.5 (D) 62.45

8. A demand paging system takes 50 time units to handle a page fault and 200 time units to replace a dirty page. Access time of memory is 2 time units. Probability of page fault and dirty page is P. Average access time is 4 time units. Then what is the value of P?

(A) 0.037 (B) 0.027
(C) 0.012 (D) 0.042

9. Assume that a total memory 20 KB is available with no partition. If Buddy system technique is used and there are total of four partitions to serve the request, the closest range of the requested size is

(A) 12.5 and 25 (B) 14.5 and 30
(C) 15.5 and 25 (D) 14.5 and 40

10. A system uses FIFO page replacement algorithm. It has three page frames with no pages loaded. First 50 pages are accessed in some order and the same pages are accessed in the reverse order. What is the number of page faults?
(A) 98 (B) 96
(C) 97 (D) 95

11. If 32-bit addressing is used for pages whose maximum size is 512 KB, what is the maximum number of pages that can be addressed?
(A) 4096 (B) 2048
(C) 8192 (D) 16384

12. Calculate the overhead due to page table if given the average process in bytes is 16-bytes, the page size is 32-bytes and the page entry is 2-bytes.
(A) 15 (B) 17
(C) 18 (D) 19

13. In a system with 32-bit virtual address and 1 KB page size, use of one-level page tables for virtual to physical address translation is not practical because of
(A) The large amount of internal fragmentation
(B) The large amount of external fragmentation
(C) The large memory overhead in maintaining page tables
(D) The large computation overhead in the translation process

14. If an instruction takes time 10 m sec if there is no page fault and time 20 m sec if there is a page fault, what is the effective instruction time if page fault occurs once every 5 instructions?
(A) 12.5 msec (B) 12 msec
(C) 14 msec (D) 15 msec

15. Let an instruction take 10 ms and page fault takes an additional 5 ms. If the average page fault occurs after 20 instructions, the effective instruction time will be
(A) 10 ms (B) 10.25 ms
(C) 0.25 ms (D) 10.75 ms

16. A 0.8 MB-sized memory is managed using variable partitions, while the rest of the memory is occupied by a 0.26 MB partition, 0.27 MB partition and 0.25 MB partitions in the order. Best-fit strategy is being adopted where would be a 0.18 MB allocation request is fulfilled?
(A) 0.26
(B) 0.27
(C) 0.25
(D) Request will be denied

Common data for questions 17 and 18: Suppose that the OS uses variable length partitions for memory management. At some particular time, the running process occupies a partition between physical addresses 20,000 and 40,000.

17. The values of base and limit register are respectively
(A) 20, 000, 40,000
(B) 20,000, 20,000
(C) 0, 10,000
(D) 0, 40,000

18. What physical address corresponds to a virtual address of 13,000?
(A) 13,000
(B) 43,000
(C) 33,000
(D) Out of range

19. Consider a page table where translation look ahead buffer is used. TLB hit ratio is 0.95 and generally takes 1 nanosecond to retrieve the frame number. If a miss is recorded by the TLB then an additional overhead of 10 nanoseconds should be taken into consideration, further cache and main memory reference takes 100 ns on average, what is the average memory fetch time using the TLB? Assume main memory accesses are always a success.
(A) 101 ns (B) 105 ns
(C) 200 ns (D) 205 ns

20. Consider a 1 MB process which is divided into five segments. Each segment is further divided into pages whose size is 4 KB. What is the maximum segments possible? Assume that the system is byte addressable.
(A) 64 (B) 128
(C) 256 (D) 512

Practice Problems 2

Directions for questions 1 to 20: Select the correct alternative from the given choices.

1. Consider a logical address space of 32 pages of 2048 words mapped into memory of 64 frames. Then the number of bits required for logical address are
(A) 16-bits (B) 17-bits
(C) 18-bits (D) 20-bits

2. In which of the page table techniques the logical address space is broken into multiple page table?
(A) Inverted Page Table
(B) Hierarchical Page Table
(C) Hashed Page Table
(D) None of the above

3. Consider a system with 70% hit ratio, 60 nanoseconds time to search the associative registers, 800 nanoseconds to access memory. What is the effective memory access time?
(A) 1200 nsec (B) 1100 nsec
(C) 1300 nsec (D) 2200 nsec

4. On a system using fixed partitions, all of size 2^8, the number of bits used by the limit register is
(A) 128 (B) 256
(C) 8 (D) 1024

5. Working set (t, k) at an instant of time, t, is the set of
(A) k future references that the operating system will make
(B) future references that the operating system will make in the next 'k' time units
(C) k references with high frequency
(D) pages that have been referenced in the last k time units.

6. Cache and interleaved memories are ways of speeding up memory access between CPUs and slower RAM. Which of the following memory models are best suited (i.e., improves performance the most) for which programs?
(i) Cached memory is best suited for small loops.
(ii) Interleaved memory is best suited for small loops.
(iii) Interleaved memory is best suited for large loops.
(iv) Cache memory is best suited for large sequential code.
(A) (i) and (ii) are true
(B) (i) and (iii) are true
(C) (iv) and (ii) are true
(D) (iv) and (iii) are true

7. A paging system with a page table in memory every reference to memory takes 100 ns. The TLB hit ratio is 85% and the time needed for searching TLB is almost negligible. What is the effective memory access time?
(A) 115 ns (B) 135 ns
(C) 145 ns (D) 125 ns

8. Consider the page sequence 4, 2, 1, 5, 3, 2, 1, 5, 0, 2, 5. If FIFO page replacement algorithm is used and frame size is 3, then the percentage of page fault is
(A) 99% (B) 90.9%
(C) 80.8% (D) 89.9%

9. If the page size is 32 KB, primary page table contains 4096 entries and the secondary page table contains 256 entries, then what is the size of logical address in bits?
(A) 15-bits (B) 20-bits
(C) 32-bits (D) 35-bits

10. If page size is 2 KB and logical address is 20-bit, then the number of entries in the page table is
(A) 2048 B (B) 256 B
(C) 512 B (D) 1 MB

11. Consider a paging system with the page table stored in memory. If a memory reference takes 200 ns, how long does a paged memory reference take?
(A) 100 ns (B) 200 ns
(C) 300 ns (D) 400 ns

12. Consider a logical address space of eight pages of 1024 words each mapped onto a physical memory of 32 frames. How many bits are there in the logical address and in the physical address?
(A) 10, 18 (B) 13, 18
(C) 13, 15 (D) 10, 5

13. For a paged system, TLB hit ratio is 0.9. Let the RAM access time 't' be 20 ns and the TLB access time 'T' be 100 ns. Then effective memory access (with TLB) will be
(A) 120 ns (B) 200 ns
(C) 130 ns (D) 150 ns

14. Assume that a user program is 100 K words and secondary storage device is a fixed hard disk with an average latency of 8 ms and a transfer rate of 2,50,000 words/second. Then find the swap time of a transfer of 100 K words to or from memory.
(A) 816 ms (B) 408 ms
(C) 204 ms (D) 8 ms

15. Consider the following segment table:

Segment	Limit	Base
0	1000	1400
1	400	6300
2	400	4300
3	1100	3200
4	1000	4700

The physical address for a logical address which is in segment 2 with offset 253 is
(A) 4553 (B) 6353
(C) 6253 (D) 4453

16. Consider a process of size 2 MB. If the page size is 0.5 KB, what is the size of the page table (assuming that each page is mapped by a 32-bit size page table entry)?
(A) 8 KB (B) 16 KB
(C) 24 KB (D) 32 MB

17. A CPU generates 32-bit virtual address. The page size is 2 KB. The translation look-aside buffer (TLB) which can hold 256 page table entries and is two-way set associative mapping. The number of bits in the TLB tag is
(A) 10-bits (B) 12-bits
(C) 14-bits (D) 15-bits

18. Assume that a total memory M is available with no partitions made yet. If Buddy system strategy is being used and a total of n partitions have been made to serve the request. The closest range of the requested size is
(A) $\frac{M}{2^n+1}$ and $\frac{M}{2^n}$ (B) $\frac{M}{n}$ and $\frac{M}{n-1}$
(C) $\frac{M}{2^n}$ and $\frac{M}{2^{n-1}}$ (D) $\frac{M}{n+1}$ and $\frac{M}{n}$

19. The memory has four free blocks of sizes 2K, 6K, 20K, 4K. The request blocks are allocated according to best fit allocation method. The allocation requests are stored in queue as shown:

Reqest no.	P_1	P_2	P_3	P_4	P_5	P_6	P_7
Reqest sizes	4 K	10 K	2 K	3 K	5 K	4 K	2 K
Usage time	1	4	2	6	3	1	8

The time at which request for P_7 will be completed is
(A) 10 unit time
(B) 14 unit time
(C) 20 unit time
(D) 15 unit time

20. A memory page containing a heavily used variable that was initialized very early and is in constant use is removed when _____ page replacement is used.
(A) LRU
(B) FIFO
(C) LFU
(D) Optimal

Previous Years' Questions

1. A processor uses 36-bit physical addresses and 32-bit virtual addresses, with a page frame size of 4 K bytes. Each page table entry is of size 4 bytes. A three-level page table is used for virtual to physical address translation, where the virtual address is used as follows: **[2008]**

- Bits 30–31 are used to index into the first level page table
- Bits 21–29 are used to index into the second level page table
- Bits 12–20 are used to index into the third level page table, and
- Bits 0–11 are used as offset within the page

The number of bits required for addressing the next level page table (or page frame) in the page table entry of the first, second and third level page tables are, respectively,
(A) 20, 20 and 20 (B) 24, 24 and 24
(C) 24, 24 and 20 (D) 25, 25 and 24

2. How many 32 K × 1 RAM chips are needed to provide a memory capacity of 256 K bytes? **[2009]**
(A) 8 (B) 32
(C) 64 (D) 128

3. In which one of the following page replacement policies, Belady's anomaly may occur? **[2009]**
(A) FIFO (B) Optimal
(C) LRU (D) MRU

4. The essential content(s) in each entry of a page table is/are **[2009]**
(A) Virtual page number
(B) Page frame number
(C) Both virtual page number and page frame number
(D) Access right information

5. A multilevel page table is preferred in comparison to a single-level page table for translating virtual address to physical address because **[2009]**
(A) It reduces the memory access time to read or write a memory location.
(B) It helps to reduce the size of page table needed to implement the virtual address space of a process.
(C) It is required by the translation look-aside buffer.
(D) It helps to reduce the number of page faults in page replacement algorithms.

6. A system uses FIFO policy for page replacement. It has four-page frames with no pages loaded to begin with. The system first accesses 100 distinct pages in some order and then accesses the same 100 pages but now in the reverse order. How many page faults will occur? **[2010]**
(A) 196 (B) 192
(C) 197 (D) 195

7. Let the page fault service time be 10 ms in a computer with average memory access time being 20 ns. If one page fault is generated for every 10^6 memory accesses, what is the effective access time for the memory? **[2011]**
(A) 21 ns (B) 30 ns
(C) 23 ns (D) 35 ns

8. Consider the virtual page reference string

1, 2, 3, 2, 4, 1, 3, 2, 4, 1

on a demand paged virtual memory system running on a computer system that has main memory size of three-page frames which are initially empty. Let LRU, FIFO and OPTIMAL denote the number of page faults under the corresponding page replacement policy. Then **[2012]**
(A) OPTIMAL< LRU < FIFO
(B) OPTIMAL < FIFO < LRU
(C) OPTIMAL = LRU
(D) OPTIMAL = FIFO

9. A RAM chip has a capacity of 1024 words of 8 bits each (1 K × 8). The number of 2 × 4 decoders with

enable line needed to construct a 16 K × 16 RAM from 1 K × 8 RAM is **[2013]**

(A) 4 (B) 5
(C) 6 (D) 7

Common data for Questions 10 and 11: A computer uses 46-bit virtual address, 32-bit physical address, and a three-level paged page table organization. The page table base register stores the base address of the first-level table (T_1), which occupies exactly one page. Each entry of T_1 stores the base address of a page of the second-level table (T_2). Each entry of T_2 stores the base address of a page of the third-level table (T_3). Each entry of T_3 stores a page table entry (PTE). The PTE is 32-bits in size. The processor used in the computer has a 1 MB 16-way set associative virtually indexed physically tagged cache. The cache block size is 64 bytes.

10. What is the size of a page in KB in this computer? **[2013]**

(A) 2 (B) 4
(C) 8 (D) 16

11. What is the minimum number of page colours needed to guarantee that no two synonyms map to different sets in the processor cache of this computer? **[2013]**

(A) 2 (B) 4
(C) 8 (D) 16

12. Assume that there are three page frames which are initially empty. If the page reference string is 1, 2, 3, 4, 2, 1, 5, 3, 2, 4, 6, the number of page faults using the *optimal replacement policy* is ———. **[2014]**

13. A computer has 20 physical page frames which contain pages numbered 101 through 120. Now a program accesses the pages numbered 1, 2, … 100 in that order, and repeats the access sequence THRICE. Which one of the following page replacement policies experiences the same number of page faults as the optimal page replacement policy for this program? **[2014]**

(A) Least-recently used (B) First-in-first-out
(C) Last-in-first-out (D) Most-recently-used

14. A system uses three page frames for storing process pages in main memory. It uses the Least Recently Used (LRU) page replacement policy. Assume that all the page frames are initially empty. What is the total number of page faults that will occur while processing the page reference string given below? **[2014]**

4, 7, 6, 1, 7, 6, 1, 2, 7, 2

15. Consider a paging hardware with a TLB. Assume that the entire page table and all the pages are in the physical memory. It takes 10 milliseconds to search the TLB and 80 milliseconds to access the physical memory. If the TLB hit ratio is 0.6, the effective memory access time (in milliseconds) is ———— **[2014]**

16. Consider a system with byte-addressable memory, 32-bit logical addresses, 4 kilobyte page size and page table entries of 4 bytes each. The size of the page table in the system in megabytes is _____. **[2015]**

17. Consider a main memory with five page frames and the following sequence of page references: 3, 8, 2, 3, 9, 1, 6, 3, 8, 9, 3, 6, 2, 1, 3. Which one of the following is true with respect to page replacement policies First In First Out (FIFO) and Least Recently Used (LRU)? **[2015]**

(A) Both incur the same number of page faults
(B) FIFO incurs 2 more page faults than LRU
(C) LRU incurs 2 more pages faults than FIFO
(D) FIFO incurs 1 more page faults than LRU

18. Consider six memory partitions of sizes 200 KB, 400 KB, 600 KB, 500 KB, 300 KB and 250 KB, where KB refers to kilobyte. These partitions need to be allotted to four processes of sizes 357 KB, 210 KB, 468 KB and 491 KB in that order. If the best fit algorithm is used, which partitions are NOT allotted to any process? **[2015]**

(A) 200 KB and 300 KB
(B) 200 KB and 250 KB
(C) 250 KB and 300 KB
(D) 300 KB and 400 KB

19. A computer system implements 8 kilobyte pages and a 32-bit physical address space. Each page table entry contains a valid bit, a dirty bit, three permission bits, and the translation. If the maximum size of the page table of a process is 24 megabytes, the length of the virtual address supported by the system is _____ bits. **[2015]**

20. Consider the following two *C* code segments. *Y* and *X* are one and two dimensional arrays of size *n* and $n \times n$ respectively, where $2 \le n \le 10$. Assume that in both code segments, elements of Y are initialized to 0 and each element *X*[*i*] [*j*] of array *X* is initialized to $i + j$. Further assume that when stored in main memory all elements of *X* are in same main memory page frame.

Code segment 1: **[2015]**

```
//initialize elements of Y to 0
//initialize elements X[i][j] of X to i+j
for (i = 0; i < n; i++)
Y[i] += X[0] [i];
```

Code Segment 2:

```
//initialize elements of Y to 0
```

```
    //initialize elements X[i] [j] of X to i + j
    for (i = 0; i < n; i++)
    Y[i] += X[i] [0];
```

Which of the following statements is/are correct?

S_1: Final contents of array Y will be same in both code segments

S_2: Elements of array X accessed inside the for loop shown in code segment 1 are contiguous in main memory

S_3: Elements of array X accessed inside the for loop shown in code segment 2 are contiguous in main memory.

(A) Only S_2 is correct
(B) Only S_3 is correct
(C) Only S_1 and S_2 are correct
(D) Only S_1 and S_3 are correct

21. Consider a computer system with 40-bit virtual addressing and page size of sixteen kilobytes. If the computer system has a one-level page table per process and each page table entry requires 48 bits, then the size of the per-process table is _____ megabytes. **[2016]**

22. Consider a computer system with ten physical page frames. The system is provided with an access sequence $(a_1, a_2, \ldots, a_{20}, a_1, a_2, \ldots a_{20})$, where each ai is a distinct virtual page number. The difference in the number of page faults between the last-in-first-out page replacement policy and the optimal page replacement policy is _____. **[2016]**

23. In which one of the following page replacement algorithms it is possible for the page fault rate to increase even when the number of allocated frames increases? **[2016]**

(A) **LRU** (Least Recently Used)
(B) **OPT** (Optimal Page Replacement)
(C) **MRU** (Most Recently Used)
(D) **FIFO** (First In First Out)

24. Recall that Belady's anomaly is that the page-fault rate may *increase* as the number of allocated frames increases. Now, consider the following statements:

S1: *Random page replacement* algorithm (where a page chosen at random is replaced) suffers from Belady's anomaly

S2: *LRU page replacement* algorithm suffers from Belady's anomaly

Which of the following is CORRECT? **[2017]**

(A) S1 is true, S2 is true
(B) S1 is true, S2 is false
(C) S1 is false, S2 is true
(D) S1 is false, S2 is false

25. Consider a process executing on an operating system that uses demand paging. The average time for a memory access in the system is M units if the corresponding memory page is available in memory and D units if the memory access causes a page fault. It has been experimentally measured that the average time taken for a memory access in the process is X units.

Which one of the following is the correct expression for the page fault rate experienced by the process? **[2018]**

(A) $(D - M)/(X - M)$ (B) $(X - M/(D - M)$
(C) $(D - X/(D - M)$ (D) $(X - M/(D - X)$

Answer Keys

Exercises

Practice Problems 1

1. B	**2.** B	**3.** D	**4.** C	**5.** C	**6.** C	**7.** B	**8.** A	**9.** A	**10.** C
11. C	**12.** B	**13.** C	**14.** B	**15.** B	**16.** C	**17.** B	**18.** C	**19.** A	**20.** C

Practice Problems 2

1. B	**2.** B	**3.** B	**4.** C	**5.** D	**6.** B	**7.** A	**8.** B	**9.** D	**10.** C
11. D	**12.** B	**13.** C	**14.** A	**15.** A	**16.** B	**17.** C	**18.** C	**19.** A	**20.** B

Previous Years' Questions

1. D	**2.** C	**3.** A	**4.** B	**5.** B	**6.** A	**7.** B	**8.** B	**9.** B	**10.** C
11. C	**12.** 7	**13.** D	**14.** 6	**15.** 122	**16.** 4	**17.** A	**18.** A	**19.** 36	**20.** C
21. 384	**22.** 1	**23.** D	**24.** B	**25.** B					