Chapter-4
Memory Management
- Ankit Pangeni DATE

Introduction

Memory management refers to the management of primary memory ie. RAM. It is one of the functions of operating system. It keeps track of each and every memory allocation & deallocation to a process. It also decides which process will get memory at what time.

Monoprogramming Vs Multiprogramming.

Monoprogramming	Multiprogramming
i) In monoprogramming, memory contains only one program at any point of time.	i) In Multiprogramming, memory contains more than one user program.
ii) Here, CPU executes the program & I/O operation is encountered, during that time CPU is idle. So, CPU is not efficiently used.	ii) Here, when one user program contains I/O operation, CPU switches to next user program. So, CPU is efficiently used.
iii) Main memory needs less space as only one program is present in main memory.	iii) Main memory needs more space as it contain more than one program in its main memory.
iv) Fixed size partition is used in monoprogramming.	iv) Both fixed & variable size partition can be used.

PAGE

DATE

v) Example: Batch processing in old computers & mobiles, old OS of computers & mobiles

Advantages
→ Allocation of memory is easy & cheap
→ Eliminates external fragmentation
→ More efficient swapping

Disadvantage
→ Larger memory access time
→ Internal fragmentation
→ Inverted page table

v) Ex: Modern OS like windows XP, 7, 8, 10 etc

Advantages
→ CPU is never idle
→ Supports multiple users
→ Resources are widely used

Disadvantages
→ Difficult to program a system
→ Tracking of all tasks is sometimes difficult to handle

DATE

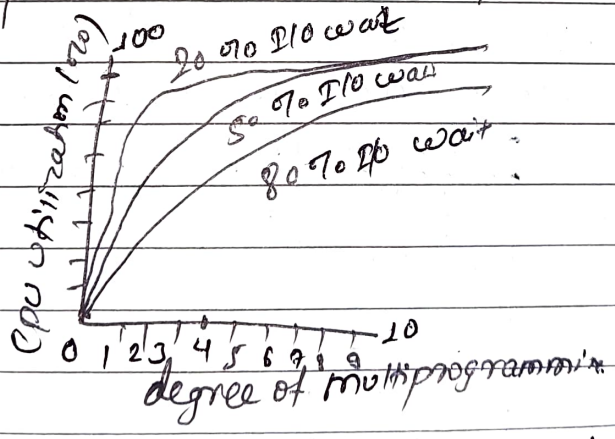
Modeling Multiprogramming.

If we have five processes that use the processor 20% of the time (spending 80% doing I/O) then we should be able to achieve 100% cpu utilization. Of course, in reality this will not happen as there may be times when all 5 processes are waiting for I/O. However, we will achieve better utilization than mono-programming.

By using the probabilities, we can build a model. Assume that a process spend p% of its time waiting for I/O. With n processes in memory, the probability that all processes are waiting for I/O (meaning the cpu is idle) is p^n.
Then, cpu utilization is given by

$$\text{cpu utilization} = 1 - p^n$$

Ex: If $p = 0.5$ & $n = 4$ then $1 - p^n = 1 - 0.5^4 = 0.9375$.



We can see that, with an I/O wait time 20%, almost 100% cpu utilization can be achieved with 4 processes. If I/O wait time is 90% then with 10 processes, we only achieve just above 60% utilization. i.e. As we use more processes, cpu utilization raises.

PAGE

DATE

* Multiprogramming with fixed & variable partitions

There are two memory management techniques. Contiguous and Non-contiguous. In contiguous, executing process must be loaded entirely in the main memory. Contiguous technique can be divided into

- Fixed (or static) partitioning
- Variable (or dynamic) partitioning.

1) Fixed partitioning

→ Dividing the main memory into a set of non-overlapping blocks is known as fixed partition.
→ This is the oldest & simplest technique used to put more than one processes in the main memory.
→ No. of partitions in RAM is fixed size but size of each partition may or may not be same.
→ No spanning is allowed and partition are made before execution or during system configure.

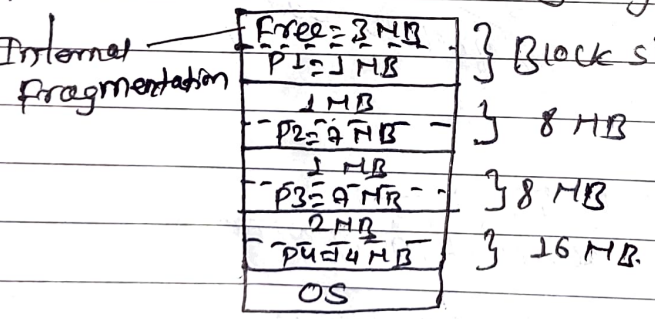


fig: Fixed partition

Here, 1st process is consuming 1MB out of 4MB in main memory. Hence, IF = 4-1 = 3MB. Sum of IF = 3+1+1+2 = 7MB

Advantages:	Disadvantage:
○ Easy to implement ○ Little OS overhead.	○ Internal Fragmentation ○ External Fragmentation ○ Limit process size. ○ Limitation on degree of multiprogram

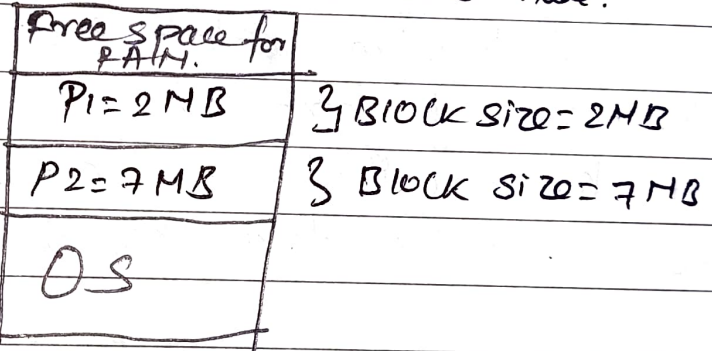
2) Variable partitioning

→ Here, when an execution request of a process has to be made, then the memory is partition according to the size needed by the processes so that there will be no internal & external fragmentation

→ When a process requested for some memory, then it will be allocated by the process so that there will be no case where some memory spaces will be left.

→ It is a memory management approach to create partition dynamically to meet the requirements of each requesting process.

→ When a process terminates, the memory manager can return the vacant space to pool of free memory area from which partition allocations are made.



→ Here the no. of partitions in RAM is not fixed & depends on the no. of incoming process & main memory size.

Advantages	Disadvantages
○ No internal fragmentation ○ No restriction on degree of multiprogramming. ○ No limitation on the size of the process	○ Difficult implementation ○ External fragmentation

DATE

Relocation and Protection.

Relocation

→ The ability to move process around in memory without affecting execution is called relocation.
→ Programmers typically do not know in advance which other programs will be resident in main memory at the time of execution of their program. Active processes need to be able to be swapped in & out of main memory in order to maximize processor utilization.
→ Memory management must convert program's logical address into physical address.
→ Address of process is at stored as virtual address.
→ Branch instructions contain the address of next instruction to be executed.
→ Data reference instructions contain the address of byte or word of data referenced.

Two types of relocation: static & dynamic.

- Static: Program must be relocated before or during loading of process into memory. Program must always be loaded into same address space in memory, or relocater must be run again.
- Dynamic: Process can be freely moved around in memory. Virtual to physical address space is done at runtime.

classmate

PAGE

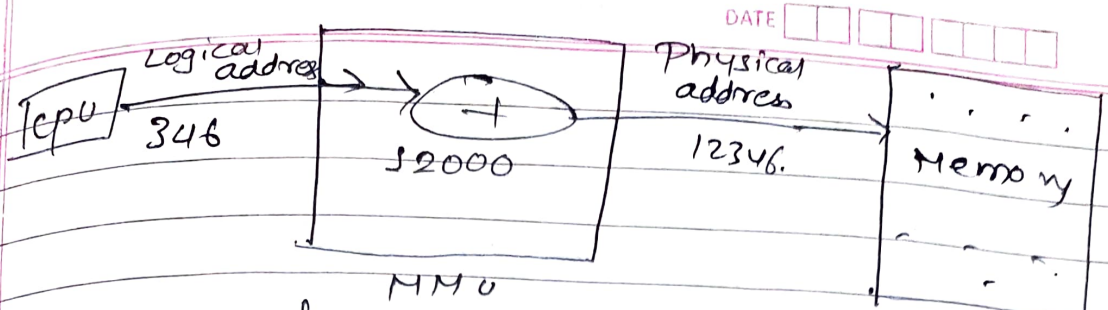


fig: Memory relocation

Protection.

→ Memory protection is a way to control memory access rights in a computer, & is a part of most modern Instruction set architectures & OS.

→ Its main purpose is to prevent a process from accessing memory that has not been allocated to it.

→ This prevents a bug or malware within a process from affecting other processes, or the OS itself.

→ Attempt to access unallocated memory results in hardware fault, called segmentation fault or storage violation exception, that causes abnormal termination of the offending process.

→ Memory protection also includes address space layout randomization & executable space protection.

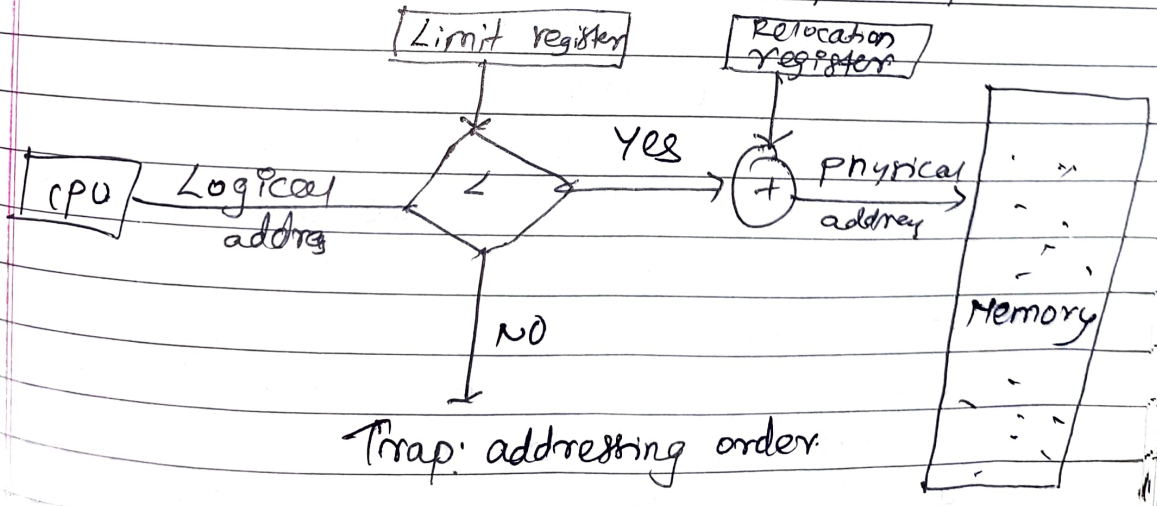


fig: Memory protection & relocation

* Space Management

DATE

* Fragmentation

As the processes are loaded and removed from the main memory, the free memory space is broken into little pieces. It happens after something that process cannot be allocated to memory block considering their small size and memory blocks remain unused, this problem is known as fragmentation.

There are two types of fragmentation: Internal & External. In internal, memory block assigned to a process is bigger. Some portion of memory is left unused, as it cannot be used by another process. In external, total memory space is enough to satisfy a request or to reside a process in it, but it is not continuous so it can't be used.

* Compaction.

Compaction is a technique by which the program are relocated in such a way that the small chunks of free memory space are made continuous to each other & clubbed together into a single free partition, that may be big enough to accomodate some more program.

By applying this technique, we can store the bigger process in the memory. The free partitions are merged which can now be allocated according to the needs of new processes. This technique is called defragmentation.

CLASSMATE

PAGE

DATE

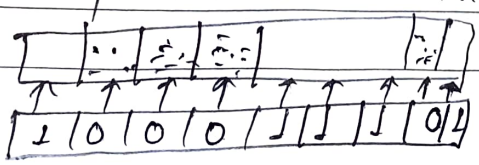
Memory Management (Bitmaps & Linked-List)

Memory management is the functionality of an OS which handles or manages primary memory and moves process back & forth between main memory & disk during execution. It keeps track of each and every memory location, regardless of either it is allocated to some process or it is free. It decides which process will get memory at what time. Tracks whenever some memory gets freed or unallocated & correspondingly updates its status. Two way of doing memory management.

1) Memory management by Bitmaps.

Under this scheme, the memory is divided into allocation units & each allocation unit has a corresponding bit in a bit map. A bit map is a collection of bits where each bit corresponds to a disk block. If the bit is zero, the memory is free. If the bit in the bit map is one, then the memory is being currently used.

The main problem is the size of allocation unit. The smaller the allocation unit, larger the bit map has to be. Another problem is when we need to allocate memory to a process. Assume allocation size is 4 bytes. If a process requests 256 bytes of memory, we must search the bit map for 64 consecutive zeros. This is slow operation so bit maps are not often used.

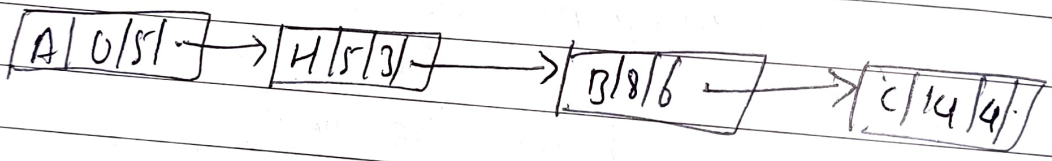


PAGE

DATE

2. Memory Management with Linked List

Another way to keep track of memory is to maintain a linked list of allocated & free memory segments, where a segment is either a process or hole between two processes. The list specifies process (P) and Hole(H) with their starting address length and a pointer to the next entry. Segment list is kept sorted by address. Sorting this way has the advantage that when a process terminates or is swapped out, updating the list is straight forward.

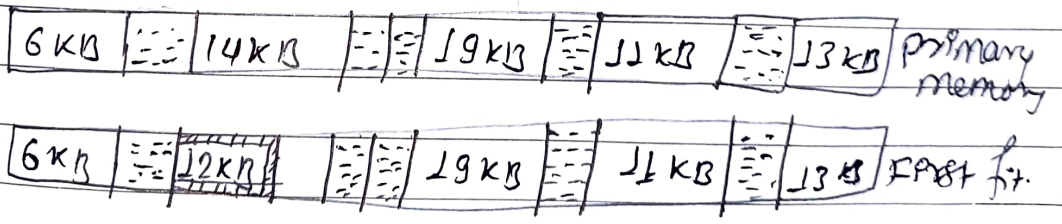


Memory Management Allocation strategies

Memory allocation strategies is the process of assigning blocks of memory on request. Sever algorithms can be used to allocate memory for a newly created process. We assume that a memory manager knows how much memory to allocate. Some popular methods used to allocate memory for a process are:

1. First Fit

This is the simplest algorithm in which the process manager scans along the list of segments until it finds a hole that is big enough. The hole is then broken up into two pieces, one for the process and one for the unused memory. It is a fast algorithm because it searches as little as possible. Ex, suppose a process request 12KB of memory and the memory manager currently has a list of unallocated blocks of 6KB, 14KB, 19KB, 11 KB & 13KB blocks. The first-fit strategy will allocate 12KB of the 14KB block to a process.



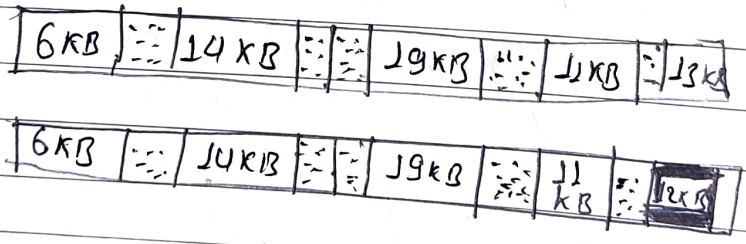
2. Next fit

It works the same way as first fit, except that it keeps track of where it is whenever it

finds a suitable hole. The next time it is called to find a hole, it starts searching the list from the place where it left off last time. Instead of always at the beginning, as first fit does.

3) Best Fit

Best fit searches the entire list and takes the smallest hole that is adequate Rather than breaking up a big hole that might be needed later, best fit tries to find a hole that is close to the actual size needed. The allocator places a process ~~req~~ in the smallest block of unallocated memory in which it will fit. Ex: suppose a process requests 12 KB memory & the memory manager currently has a list of unallocated blocks of 6 KB, 14 KB, 19 KB, 11 KB & 13 KB blocks. The best fit strategy will allocated 12 KB of the 13 KB block to the process.

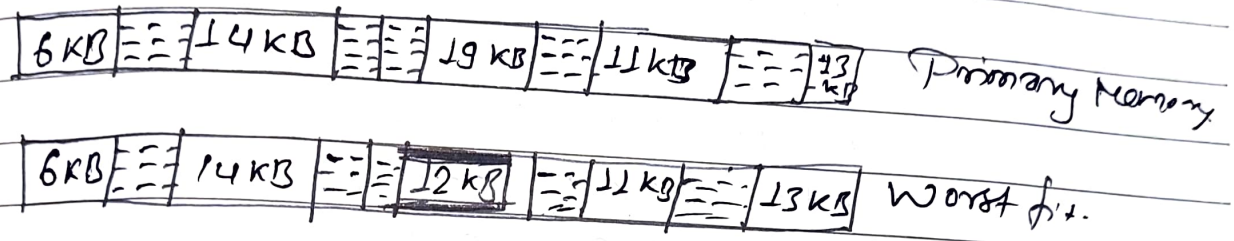


4) Worst Fit

It always takes the largest available hole, so that the hole broken off will be big enough to be useful. The memory manager places a process in the largest block of unallocated memory

DATE

available. The idea idea is that this placement will create the largest hold after the allocations, thus increasing the possibility that compared to best fit another process can use the remaining space.
Ex:- in example below, worst fit will allocate 12kB of the 19 kB block to the process, leaving a 7kB block for future use.



5) Quick fit

This algorithm maintains a separate list for the common sizes requested. Ex, it might have a table with n entries, in which the first entry is a pointer to the head of a list of 4-kB holes; the second entry is a pointer to a list of 8-kB holes, the third entry to a pointer of 12-kB hole and so on. Holes of say, 21 kB, could either be put on the 20-kB list or a special list of odd sized hole.
With quick fit, finding a hole of the required size is extremely fast.
Its disadvantage is that, if a process requesting larger memory arrives at a later stage then it cannot be accomodated as the largest hole is already split & occupied.

PAGE

DATE

Numerical problem

Given memory partitions of 100k, 500k, 200k, 300k, and 600k (in order) how would each of the First-fit, Best-fit, and Worst-fit algorithms place processes of 212k, 417k, 112k, & 426k (in order)? Which algorithm makes the most efficient use of memory?

=> For First-fit algorithm.

1. 212k is put in 500k partition
2. 417k is put in 600k partition
3. 112k is put in 288k partition (new partition = 500-212=288k)
4. 426k is put in nowhere, it must wait.

For Best-fit algorithm

1. 212k is put in 300k partition
2. 417k is put in 500k partition
3. 112k is put in 200k partition
4. 426k is put in 600k partition

For Worst-fit algorithm

1. 212k is put in 600k partition.
2. 417k is put in 500k partition
3. 112k is put in 388k partition (600-212)
4. 426k is put in no where, it must wait

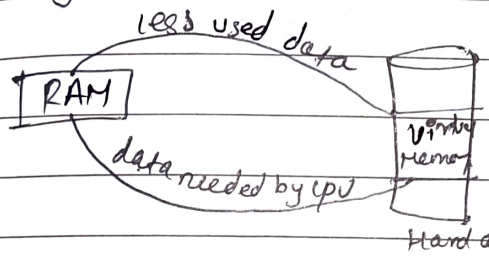
We can see, Best-fit turns out to be the best.

* Virtual Memory

DATE

Virtual memory is a technique of executing programs or instructions that may not fit entirely in the system's memory. It is implemented with the help of secondary storage device by transferring data from the main memory to secondary memory & vice-versa.

Its main objective is to provide multiprogramming. Its use has its limitations, particularly with speed. It's generally better to have as much physical memory as possible so programs work directly from RAM or physical memory. Its use also slows a computer because data must be mapped between virtual & physical memory, which requires extra hardware support.



(logical)

* Virtual address :- The address generated by processor when a program is running is known as logical address. The logical address is virtual as it doesn't exist physically. It is used as reference to access the physical memory location. set of all logical addresses generated by a programs perspective is called Logical address space.

* Physical address :- It identifies a physical location in a memory. The logical address is mapped into its corresponding physical address by a hardware device called memory Management unit (MMU). The user

classmate

PAGE

DATE

never deals with physical address. Instead, the physical address is accessed by its corresponding logical address by the user. The user program generates the logical address & thinks that the program is running in logical address. But the program needs physical memory for its execution. Hence, the logical address must be ~~conver~~ mapped into physical address before they are used.

The set of all physical address corresponding to the logical addresses in a logical address space is called Physical address space.

	Logical address	physical address
i)	It is generated by CPU in perspective of a program.	i) It is a location that exists in the memory unit.
ii)	Set of all logical addresses generated by CPU for a program is called logical address space.	ii) Set of all physical addresses mapped to corresponding logical addresses is called physical address space.
iii)	Also called virtual address as it doesn't exist physically in the memory unit.	iii) Can be accessed physically
iv)	Logical address is generated by CPU while program is running	iv) Physical address is computed by MMU (Memory Management unit)

classmate

PAGE

DATE

Swapping

Swapping is a mechanism in which a process can be swapped temporarily out of main memory (or move) to secondary storage (disk) and make that memory available to other processes. At some later time, the system swaps back the process from the secondary storage to main memory. For improving the performance of system, we use the concept of swapping.
When there is a situation to perform swapping, we use the sweeper for;

- selecting which process to be out
- selecting which process to be in
- providing memory space to the processes those are newly entered.

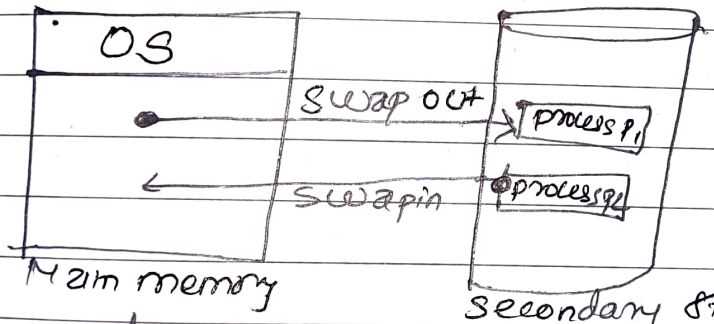


fig: swapping of two processes.

Paging

Paging is a memory management scheme by which a computer stores and retrieves data from secondary storage for use in main memory. In this scheme, the OS retrives data from secondary

PAGE

classmate

DATE

Storage in same-size blocks called pages (size is power of 2, between 512 & 8192 bytes). Paging technique plays an important role in implementing virtual memory. The size of the process is measured in number of pages. Similarly, main memory is divided into small fixed-sized blocks of (physical) memory called frames and size of a frame is kept same as that of a page to have optimum utilization of main memory and to avoid external fragmentation.

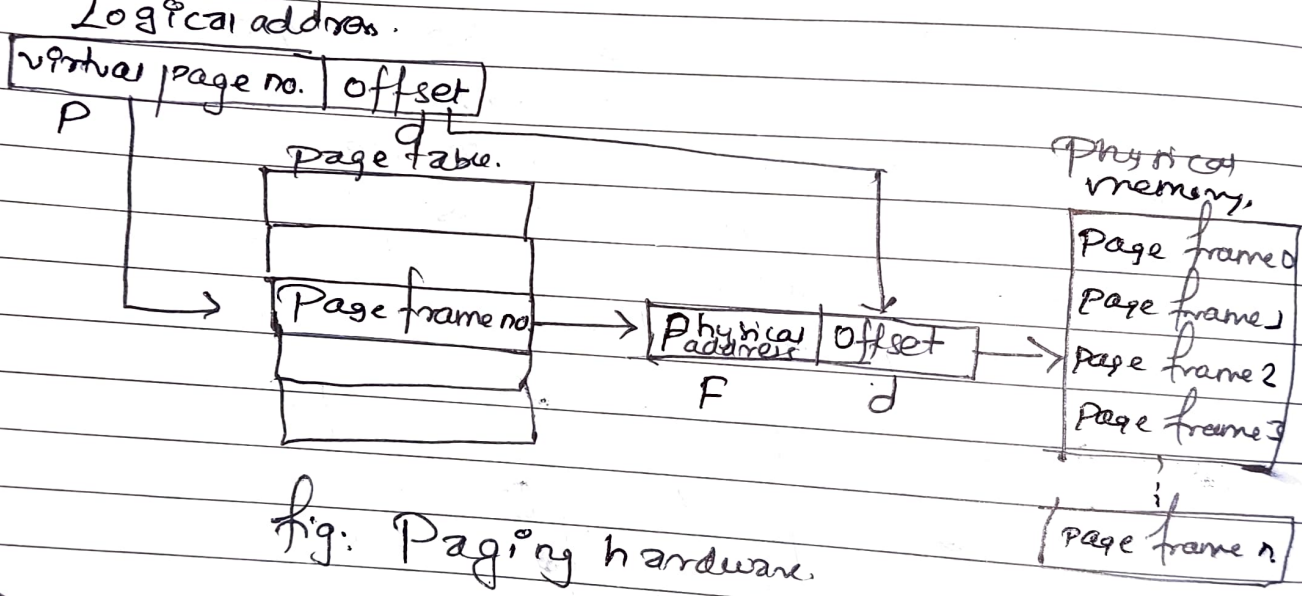


fig: Paging hardware.

Draw backs of paging

→ Size of Page table can be very big and therefore it wastes main memory.
→ CPU will take more time to read a single word from the main memory.

Page Table.

It is a data structure used by virtual memory system in a computer OS to store the mapping between virtual and physical addresses. It holds information such as page number, offset, present/absent bit, modified bit etc. It is a function with the virtual page number as argument and the physical page number as result.

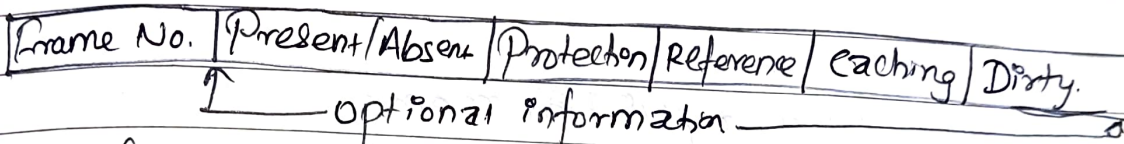


fig: Page table entry structure.

* Frame No: It gives the frame no. value in which the current page will be mapping.
No. of bits for frame = $\frac{\text{size of physical memory}}{\text{Frame size}}$
* Present/Absent bit: Specifies whether a particular page you are looking for is present or absent. In case if it's not present, that is called page fault
* Protection bit: Identifies the kind of protection such as read, write, etc.
* Referenced bit: It will say whether this page has been referred in the last clock cycle or not. It is set to 1 by hardware when page is accessed.
* Caching bit: It identifies whether the data is fresh or not.
* Dirty bit: says whether the page has been modified or not. Modified means sometimes you try to write something on to the page. It is set to 1 by hardware on write-access to page.

DATE

Note:
The no. of bits required depends on the number of frames. Number of bits for frame = size of physical memory / frame size.

Virtual address space = size of process
Process size = 2^x bytes, no. of bits in virtual address space = x bits frame size = page size.

Page number = virtual memory / page size.

Q. Consider a virtual memory and physical memory of size 128 MB and 32 MB respectively. Assume that page size is 4k, what will be the number of bits required for the page number, frame no. & offset?

⇒ Virtual memory = 128 Mb = 2^7
Physical memory = 32 Mb = 2^5
Page size = 4k = 2^2, So 2 number of bits are required for offset.

∴ Page no = $\frac{2^7}{2^2} = 2^5$

∴ 5 no. of bits are required for page number.

Frame size = page size = 2^2

∴ Frame size = page size = 2^2

∴ Frame no. = $\frac{\text{physical memory}}{\text{frame size}} = \frac{2^5}{2^2} = 2^3$

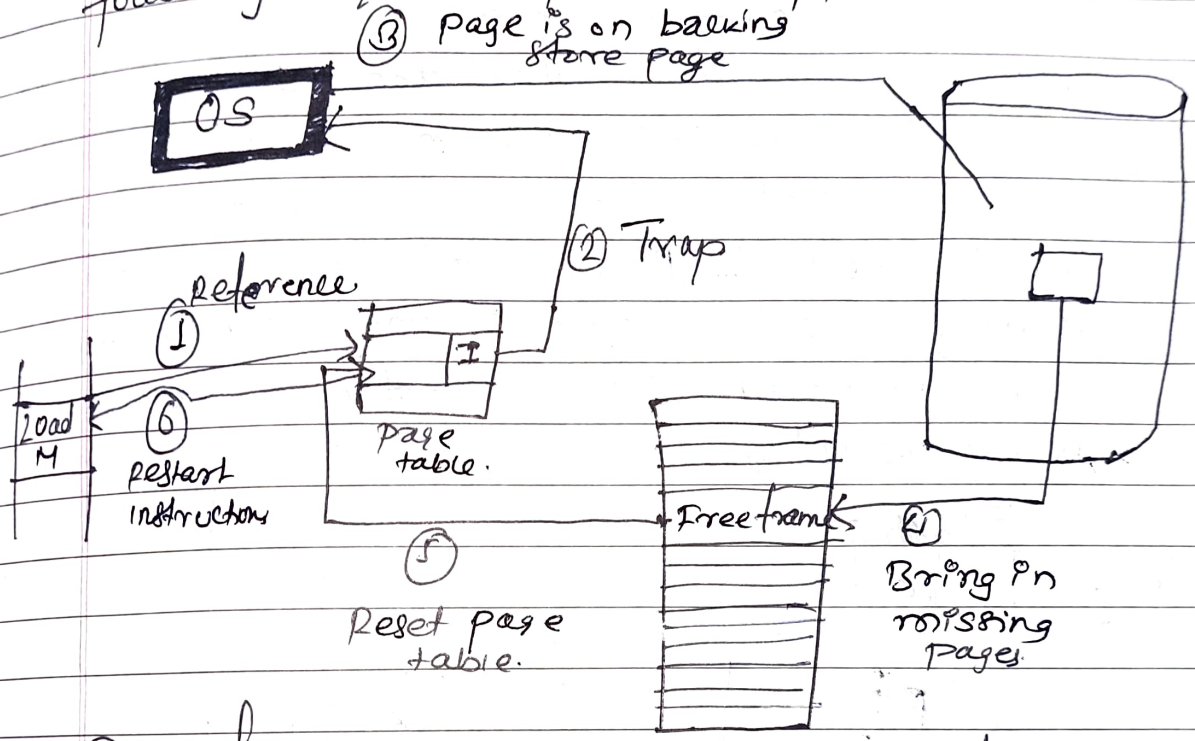
so, 3 bits are req^d for frame number

classmate PAGE

DATE

Handling Page Faults.

A page fault occurs when a program attempts to access data or code that is in its address space, but is not currently located in the system RAM. So when the page fault occurs, the following sequence of event happens.



Page fault can be handled in series of steps.

→ The memory address requested is checked 1st to make sure it was a valid memory request.

→ If the sequence was invalid, the process is terminated otherwise the page must be in.

→ A free frame is located from a free-frame list

→ A disk operation is performed to bring the necessary page from disk.

→ The process page table is updated with new frame no. and indicate a valid page reference.

→ The instruction that may cause the page fault will be restarted from the beginning.

PAGE

DATE

Translation Lookaside Buffer (TLB)

high speed.

It is a temporary cache memory to keep track of recently used transition. It contains page table that have been most recently used. Given a virtual address, processor examines the TLB If page table entry is present (TLB hit), the frame number is retrieved and the real address is formed. If the page table entry is not found in the TLB (TLB miss), the page number is used to index the process page table. TLB 1st checks if page is already in main memory, if not in main memory a page fault is issued then the TLB is updated to include the new page entry.

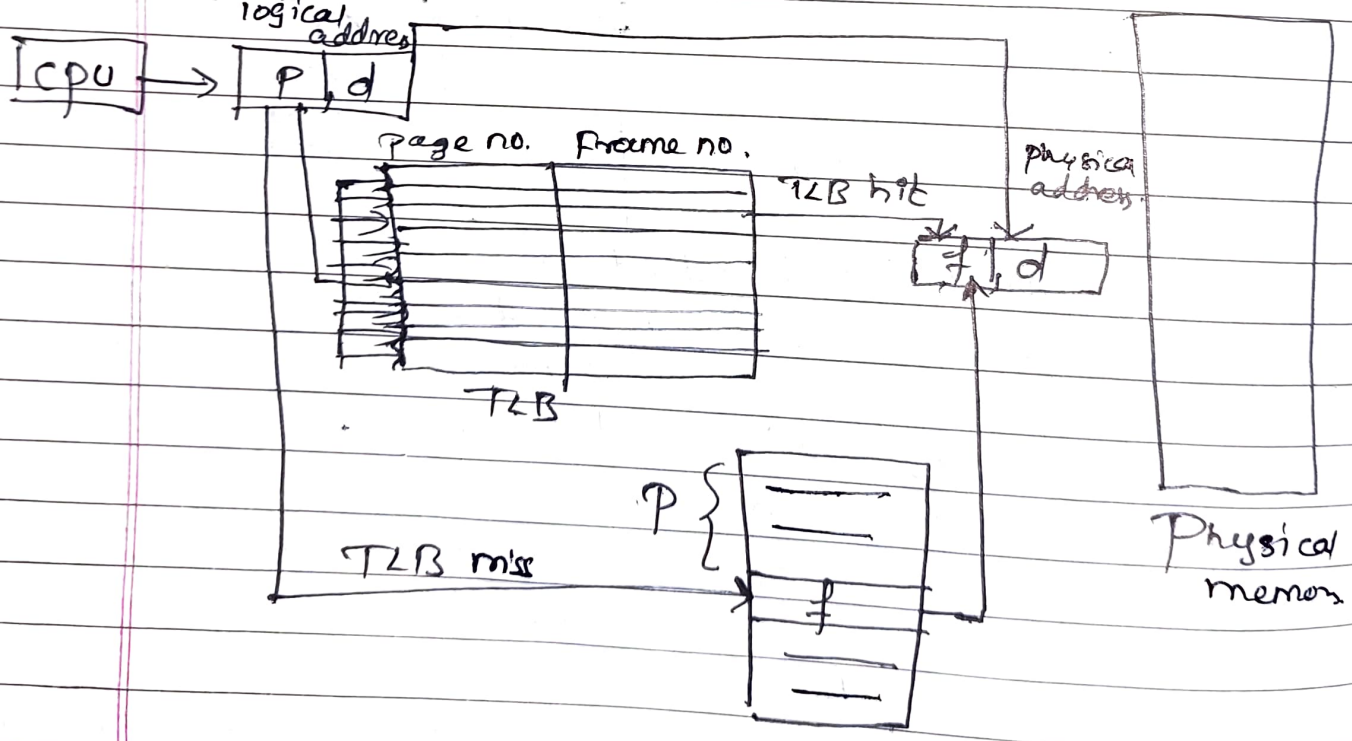


fig: Paging hardware with TLB

PAGE

DATE

Page Replacement Algorithms

A page replacement algorithm is needed to decide which page needs to be replaced when new page comes in. Since, physical memory is much smaller than virtual memory, page faults happen In case of page fault, OS might have to replace one of the existing pages with newly needed page. Different page replacement algorithms suggest different ways to decide which page to replace. The target for all algorithms is to reduce the number of page faults.

Page fault: It happes when a running program accesses a memory page that is mapped into the virtual address space, but not loaded in physical memory

1) FIFO Page Replacement Algorithm

This is the simplest page replacement algorithm. OS keeps track of all pages in the memory in a queue; oldest page is in the front of the queue. When a page needs to be replaced, page in the front of the queue is selected for removal.

Eg. Given reference string:

7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 1, 2, 0

f3			1	1	1	1	0	0	0	3	3	3	3	2	2		
f2		0	0	0	0	3	3	3	2	2	2	2	1	1	1		
f1	7	7	7	2	2	2	2	4	4	4	0	0	0	0	0		
	*	*	*	*	Hit	*	*	*	*	*	*	Hit	*	*	Hit		

CLASSMATE

PAGE

DATE

Total

∴ Hits = 3

∴ Page fault / page miss = 12

Hit ratio = $\frac{\text{no of hits}}{\text{total references}} = \frac{3}{15} \times 100 = 20\%$

Miss ratio = $\frac{12}{15} \times 100 = 80\%$

Note: Hit : If the requested page is already present in the main memory then we call it as hit.

Advantages of FIFO

→ Easy to understand and program
→ Distribute fair chance to all.

Disadvantages

→ It is likely to replace heavily used pages if they are still needed for further processing
→ System needs to keep track of each frame.

2) Belady's anomaly

In computer storage, Belady's anomaly is the phenomenon in which increasing the number of page frames results in an increase in the number of page faults for certain memory access patterns. This phenomenon is commonly

classmate

PAGE

experienced when using the FIFO page rep. algo.

Example: use 4 frames in above example and observe that the no. of page fault increases

3) Second Chance page replacement algorithm

It is a modified form of the FIFO page replacement algorithm. One way to implement it is to have a circular queue. If the value is equal to 0, then we proceed to replace the page. But if the reference bit is equal to 1, then we give page the second chance. When the page gets second chance its reference bit is cleared.

Ex: Consider reference string 2,3,2,1,5,2,4,5,3,2,5,2 and no. of frames allocated = 3, we get 7 total page faults by using this page replacement algorithm

	2	3	2	1	5	2	4	5	3	2	5	2
f_0				1(0)	1(0)	1(1)	4(0)	4(0)	4(0)	2(0)	2(0)	2(1)
f_1		3(0)	3(0)	3(0)	5(0)	5(0)	5(0)	5(1)	5(0)	5(0)	5(1)	5(1)
f_2	2(0)	2(0)	2(1)	2(1)	2(0)	2(0)	2(0)	2(0)	3(0)	3(0)	3(0)	3(0)
	X	X	Hit	*	*	Hit	*	Hit	X	X	Hit	Hit

Total page faults = 7

Advantages: 1) Obvious improvement over FIFO
2) Allows commonly used pages to stay in queue.

Disadvantages: 1) Still suffers from Belady's anamoly.

4) Optimal page replacement algorithm

Here, whenever the page fault occurs, we replace the page which would not be used for the longest duration of time in near future. It has the lowest page fault rate among all the algorithms and it will never suffer from Belady's anomaly

Example: Given reference string:
7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1

	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
f_4				2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
f_3			1	1	1	1	1	4	4	4	4	4	4	1	1	1	1	1	1	1
f_2		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
f_1	7	7	7	7	7	3	3	3	3	3	3	3	3	3	3	3	3	7	7	7
	*	*	*	*	Hit	*	Hit	*	Hit	Hit	Hit	Hit	Hit	*	Hit	Hit	Hit	*	Hit	Hit

∴ Total page hits = 12

∴ Total page faults = 8

Hit ratio = $\frac{12}{20} \times 100$ %

Page miss ratio = $\frac{8}{20} \times 100$ %

Advantages: i) Less complex & easy to implement
ii) Simple data structures are used

Disadvantages:
i) Error detection is hard
ii) Not all OS can implement this

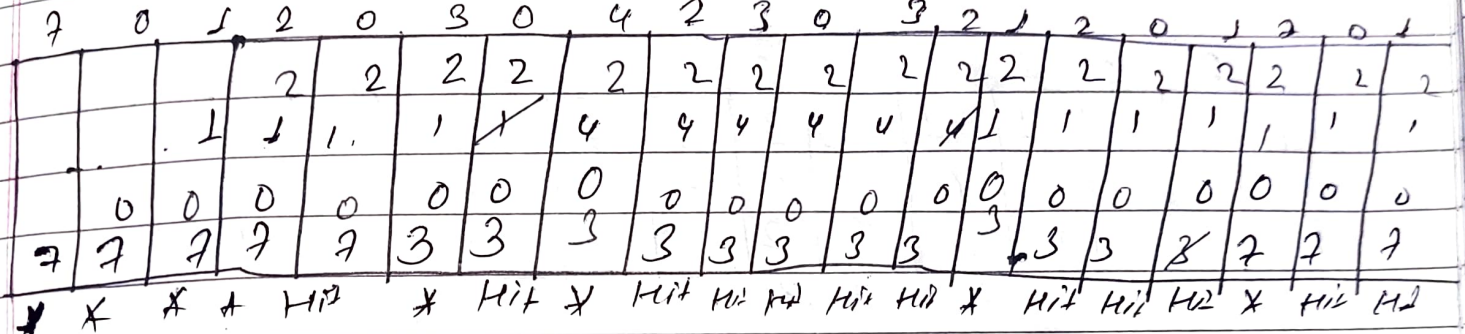
5) Least recently used (LRU) page replacement algorithm

In this algorithm, the page that has not been used for the longest period of time has to be replaced. In LRU, we check the

If all the odd pages are having same frequency then take FIFO method for that and remove the page. This page replacement algo. looks backward in time, rather than forward.

Ex: Reference String:

7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1.



∴ Total page hits = 12

Total page fault = 8

Advantages	Disadvantages
i) Amenable to full statistical analysis	i) It is slow as it always uses searching algorithm to go to past.
ii) Never suffers from Belady's anamoly.	

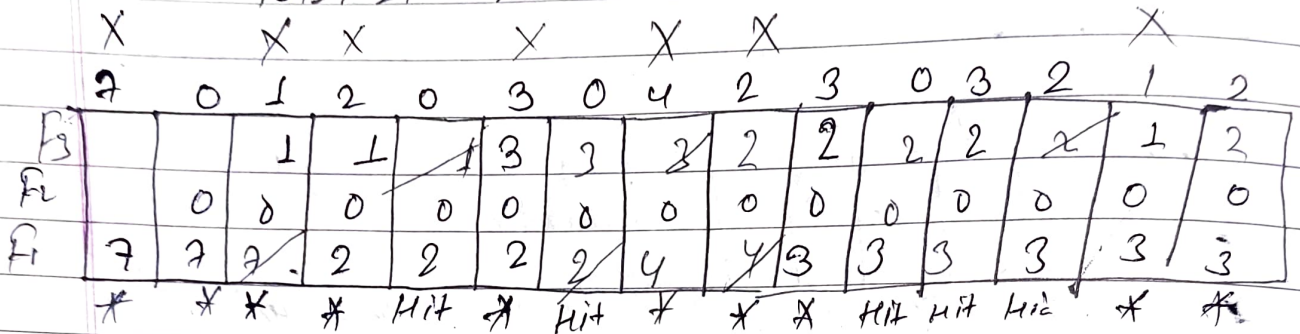
6) Least frequently used (LFU) page replacement algo

This algorithm selects a page for replacement if the page has not been used often in the past or replace page that has the smallest count. In LFU, we check the old page as well as the frequency of that page & if the frequency of the page is

DATE

larger than the old page, we cannot remove it and if all the old pages are having same frequency, then take FIFO method & remove that page.

Ex: Reference String:
7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2



Frequencies:

7 = Ø, 1, 0 | 0 = Ø, 1, 2 (4) | 1 = Ø, 1 (0) | 2 = Ø, 1 (Ø, 1 / 2, 1) | 3 = Ø, 1, Ø (1, 2) | 4 = Ø, 1, 0

Total Hit = 5
Total Miss = 10

7) Clock Page Replacement Algorithm

Clock is a more efficient version of FIFO than second-chance because pages don't have to be constantly pushed to the back of the list, but it performs the same general function as second chance.

PAGE

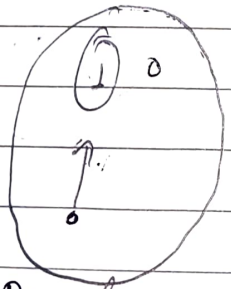
DATE

It keeps a circular list of pages in memory, with the "hand" pointing to the last examined page frame in the list.

When a page fault occurs and no empty frames exist, then the R (referenced) bit is inspected at the hand's location. If R is 0, the new page is put in the place of the page the "hand" points to, and the hand is advanced one position. O/W, the R bit is cleared, then the clock hand is incremented & the process is repeated until a page is replaced.

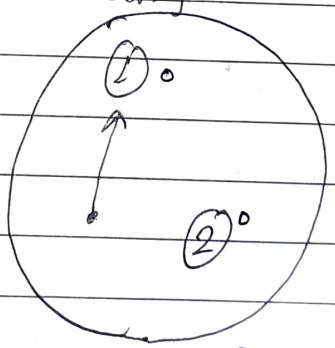
Ex: Input reference: {1, 2, 3, 4, 3, 1, 2, 1, 5, 4}
Size of Memory: 3

=) Upon accessing 1



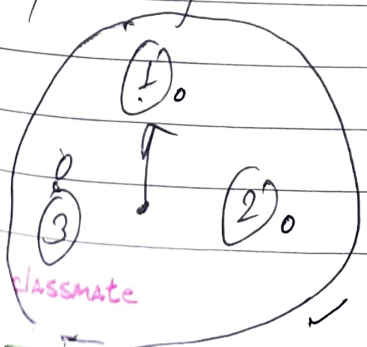
Page fault ✓

Upon accessing 2.



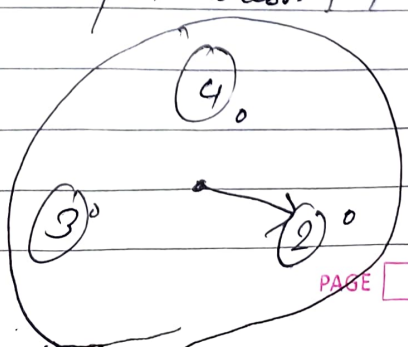
✓ page fault

Upon accessing 3



✓

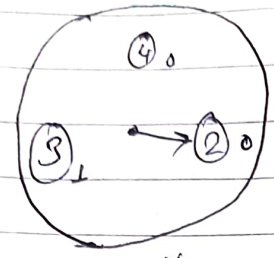
Upon accessing 4



✓

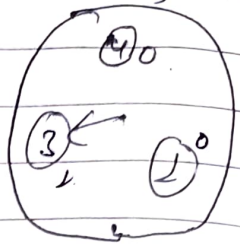
DATE

upon accessing 3



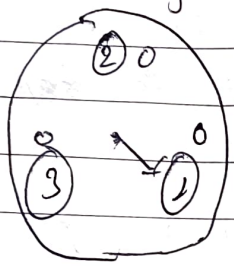
X

upon accessing 1



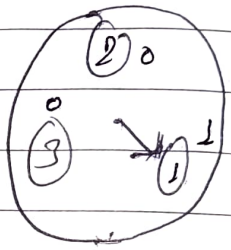
✓

upon accessing 2



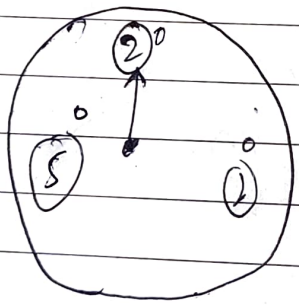
✓

upon accessing 1



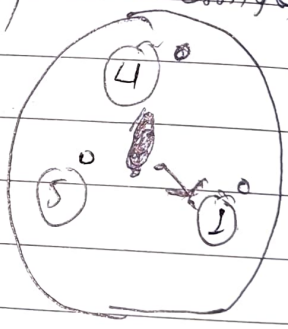
X

upon accessing 5



✓

upon accessing 4



✓

DATE

* Locality of reference.

Locality of reference is the tendency of a processor to access the same set of memory locations repetively over a short period of time. Subroutines tend to localize the references to memory for fetching instructions, this is the reason for this property. There are two basic types of locality of reference

i) Temporal locality:-

The information which is used currently is likely to be used in near future. For eg. Reuse of information in loops.

ii) Spatial Locality:

If a word is accessed, adjacent (near) words are likely to be accessed soon. Ex: Related data items (arrays) are usually stored together, Instructions are executed sequentially.

Segmentation

DATE

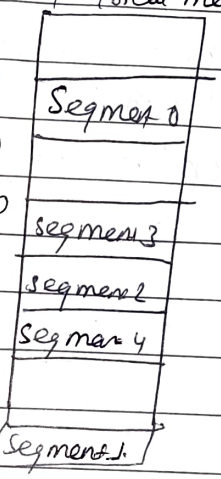
Segmentation is a memory management technique in which the memory is divided into variable sized segment. Each segment is allocated to each process. The details about each segment are stored in a table called as segment table. It contains following information about the segment:-

- Base: It is the base address of segment.
- Limit: It is the length of the segment.

The user specifies each address by two quantities Segment number & offset.

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

Segment table



Advantages:

- No internal fragmentation, Less overhead.
- Easier to reallocate segment
- Segment table is less size than page table

Disadvantage.

- It is costly memory management technique.
- Sometimes it may have external fragmentation.
- difficult to allocate contigious memory to variable sized partition.

CLASSMATE PAGE

DATE

Why Segmentation?

=) Segmentation is a memory management technique. We have different types of memory allocation and management schemes in OS. If we talk about the non-continuous memory allocation technique, we have two options: Paging & Segmentation.

Paging divides the memory into some fixed size block on the other hand segmentation divides the user program & the secondary memory into uneven-sized blocks called as segments or sections. Segmentation gives user's view of the process which paging doesn't give. Here the user's view is mapped to physical memory. (Further on you can write its advantages)

Differences

Paging	Segmentation
i) Page is always of fixed block size.	i) Segment is of variable size.
ii) Paging may lead to internal fragmentation as page is of fixed size block.	ii) It may lead to external fragmentation as the memory is fued with the variable sized blocks.
iii) In paging, user only provides a single integer as the address which is divided by hardware into a page number & offset.	iii) The user specifies the address into two quantities. ie. Segment no. and offset.

PAGE

DATE

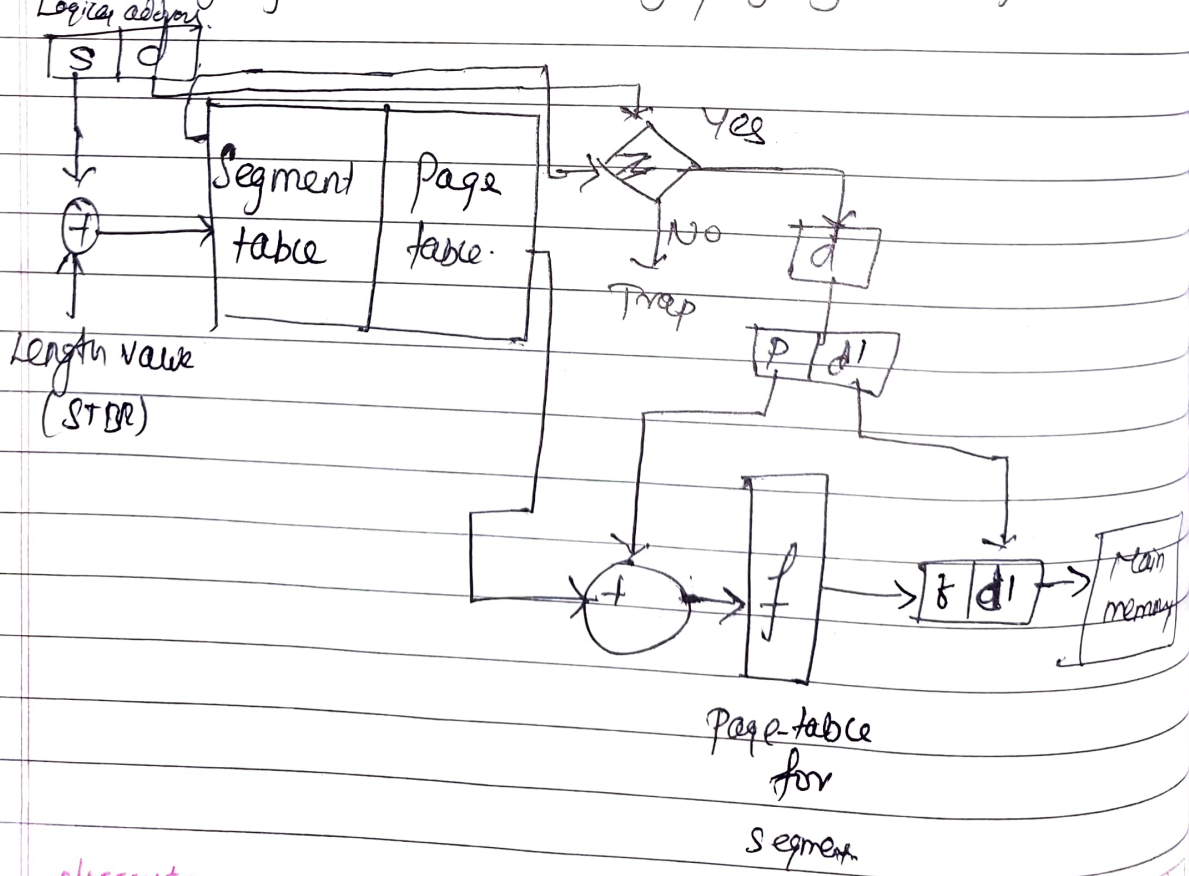
(iv) The size of the page is decided or specified by the hardware.	iv) The size of the segment is specified by the user.
v) It is faster in comparison of Segmentation.	v) It is slow.
vi) Logical address is split into page number & page offset.	vi) Logical address is split into section number & section offset.
vii) Paging is invisible to the user.	vii) Segmentation is visible to the user
viii) A programmer cannot efficiently handles data structure.	viii) It can efficiently handles data structures

DATE

* Segmentation with paging. (MULTICS)

It is a memory management technique with an implementation of a virtual memory on a system using segmentation with paging. Usually only moves individual packages back and forth between main memory & secondary storage, similar to a paged non-segmented system. Pages of segment can be located anywhere in main memory & need not be contiguous. This usually results in a reduced amount of input/output between primary & secondary storage & reduced memory fragmentation.

The MULTICS OS used Segmentation with paging. That solved problems of external fragmentation and lengthy search times by paging the segments.



classmate

PAGE

Questions asked from this Chapter

Q. Why OPR is best but not practically feasible page replacement algorithm? Calculate the number of page faults for OPR, LRU, and clock page replacement algorithm for the reference string 1,3,4, 2,3,5,4, 3,1,2, 4,6,3,2, 1,4,2. Assume that memory size is 3. (2078 - 10 marks)

Q. Differentiate between internal and external fragmentation. Suppose that we have memory of 1000 KB with 5 partitions size of 150 KB, 200 KB, 250 KB, 100 KB, and 300 KB. Where the processes A and B of size 175 KB and 125 KB will be loaded, if we used Best-Fit, and Worst-Fit Strategy? (2078 - 5 marks) (2071 - 6 marks), (2072 - 6 marks) (2068 - 6 marks)

Q. How Second Chance replacement algorithm differs from FIFO page replacement policy? Discuss the concept of Belady's anomaly with suitable example. (2076 - 10 mark)

Q. Why program relocation & protection is important? Explain the technique of achieving program relocation & protection. (2076 - 5 marks)

Q What is swapping? Differentiate contiguous memory allocation with non contiguous memory allocation. (2075 - 5 marks)

DATE

Q. Difference between physical & virtual address? Explain the conversion of virtual address into Physical address by MMU. (2074 - 6 marks) (2066 - 6 marks)

Q. Difference between paging and segmentation? Why many systems use the combination of both? (2069 - 6 marks) (2070 - 6 marks) (2071 - 6 marks)

Q. What is the virtual memory? What are the functions of associative memory? (2075 - 6 marks)

Q. Differentiate between multiprogramming and Monoprogramming. What will be the CPU utilization with 6 process with 60 % IO waiting time are in memory? (2076 - 6 marks)

Q. Consider the following reference string:
1, 2, 3, 4, 2, 1, 5, 6, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6. How many page fault would occur for LRU replacement, FIFO replacement and Optimal replacement algorithm. Assuming 3, 5, or 7 frames? Remember all frames are initially empty, so your 1st unique pages will cost one fault each. (2067 - 10 marks). (2073 - 6 marks) (2072 - 6 marks).

Q. Explain bitmap & linked list memory management system. (2074 - 6 marks) ? How much space is required in memory to store bitmap for 20 GB harddisk with 2 KB block size. (2069 - 6 marks)