USN : _____

CMR Institute of Technology, Bangalore
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
III - INTERNAL ASSESSMENT

Semester: 4-CBCS 2018 Date: 29 Jul 2021
Subject: OPERATING SYSTEMS (18CS43)
Faculty: Dr Prem Kumar Ramesh
Time: 01:00 PM - 02:30 PM Max Marks: 50

Instructions to Students:

Question 4 is compulsory & Answer Any Four Complete Questions from the remaining.

ANSWER ANY 5 Question(s)

Marks CO BT/CL

1. Define Deadlock. Write Short notes on the 4 necessary conditions that arise deadlocks

A process requests resources, if the resources are not available at that time, the process enters a waiting state. Sometimes, a waiting process is never again able to change state, because the resources it has requested are held by other waiting processes. This situation is called a**Deadlock.** **[2mrks]**

To illustrate a deadlocked state, consider a system with three CD RW drives. [2 mrks] Suppose each of three processes holds one of these CD RW drives. If each process nowrequests another drive, the three processes will be in a deadlocked state. Each is waiting for the event "CD RW is released," which can be caused only by one of theother waiting processes. This example illustrates a deadlock involving the same resource type.

Deadlocks may also involve different resource types. For example, consider a system with one printer and one DVD drive. Suppose that process Pi is holding the DVD and process P_j is holding the printer. If P_i requests the printer and P_j requests the DVD drive, a deadlock occurs.

Necessary Conditions

A deadlock situation can arise if the following four conditions hold simultaneously in a system: [6mrks]

1. **Mutual exclusion:** At least one resource must be held in a non-sharable mode,

that is, only one process at a time can use the resource. If another process requests that resource, the requesting process must be delayed until the resource has been released.

2. **Hold and wait:** A process must be holding at least one resource and waiting to acquire additional resources that are currently being held by other processes.
3. **No preemption:** Resources cannot be preempted; that is, a resource can be released only voluntarily by the process holding it, after that process has completed its task.
4. **Circular wait:** A set $\{P_0, P_l, \ldots, P_n\}$ of waiting processes must exist such that P_o is waiting for a resource held by P_1, P_1 is waiting for a resource held by P_2, ... , P_{n-1} is waiting for a resource held by P_n and P_n is waiting for a resource held by P_o.

2. Assume that there are 5 processes P0 through P5 and 4 types of resources. At time T0 we have the following state.

Process	Allocation				MAX				Available			
P0	A	B	C	D	A	B	C	D	A	B	C	D
P1	0	0	1	2	0	0	1	2	1	5	2	0
P2	1	0	0	0	1	7	5	0				
P3	1	3	5	4	2	3	5	6				
P4	0	6	3	2	0	6	5	2				
P5	0	0	1	4	0	6	5	6				

Apply Bankers algorithm to answer the following.

i) What is the content of need matrix.

ii) Is the system in a safe state.

iii) If a request from a process P1(0,4,2,0) arrives can it be granted.

Solution (a) **[10.0] 1 [3]**

What is the content of the matrix *Need*?

Need

	A	B	C	D
P_0	0	0	0	0
P_1	0	7	5	0
P_2	1	0	0	2
P_3	0	0	2	0
P_4	0	6	4	2

Solution (b)

Is the system in a safe state?

Yes, there exist several sequences that satisfy safety requirements (e.g. P_0, P_2, P_1, P_3, P_4).

Solution (c)

If a request from process P_1 arrives for (0, 4, 2, 0), can the request be granted immediately?

Pretend that the allocation can be made since the Available matrix is (1, 5, 2, 0), and it will now change to (1, 1, 0, 0). The next step is to find the safe sequence of processes. Alloc for P_1 becomes (1,4,2,0) and Need for P_1 becomes (0, 3, 3, 0).

One possible safe sequence is: P_0, P_2, P_3, P_1, P_4.

3a. Describe with an example the internal and external fragmentation problem encountered in contiguous memory allocation.**[5.0] 1 [2]**

Two types of memory fragmentation:

1. Internal fragmentation
2. External fragmentation

1. Internal Fragmentation

- The general approach is to break the physical-memory into fixed-sized blocks andallocate memory in units based on block size.
- The allocated-memory to a process may be slightly larger than the requested-memory.
- The difference between requested-memory and allocated-memory is called internalfragmentation i.e. Unused memory that is internal to a partition.

2. External Fragmentation

- External fragmentation occurs when there is enough total memory-space to satisfy a request but the available-spaces are not contiguous. (i.e. storage is fragmented into alarge number of small holes).
- Both the first-fit and best-fit strategies for memory-allocation suffer from external fragmentation.
- Statistical analysis of first-fit reveals that given N allocated blocks, another 0.5 N blocks will be lost to fragmentation. This property is known as the 50-percent rule.

Two solutions to external fragmentation:

- Compaction: The goal is to shuffle the memory-contents to place all free memory together in one large hole. Compaction is possible only if relocation is dynamic and doneat execution-time
- Permit the logical-address space of the processes to be non-contiguous. This allows a process to be allocated physical-memory wherever such memory is available. Twotechniques achieve this solution: 1) Paging and 2) Segmentation.

Paging

- Paging is a memory-management scheme.
- This permits the physical-address space of a process to be non-contiguous.
- This also solves the considerable problem of fitting memory-chunks of varying sizesonto the backing-store.
- Traditionally: Support for paging has been handled by hardware.
- Recent designs: The hardware & OS are closely integrated.

Basic Method of Paging

- The basic method for implementing paging involves breaking physical memory into fixed-sized blocks called frames and breaking logical memory into blocks of the same size called pages.
- When a process is to be executed, its pages are loaded into any available memory framesfrom the backing store.

The backing store is divided into fixed-sized blocks that are of the same size as the memory frames

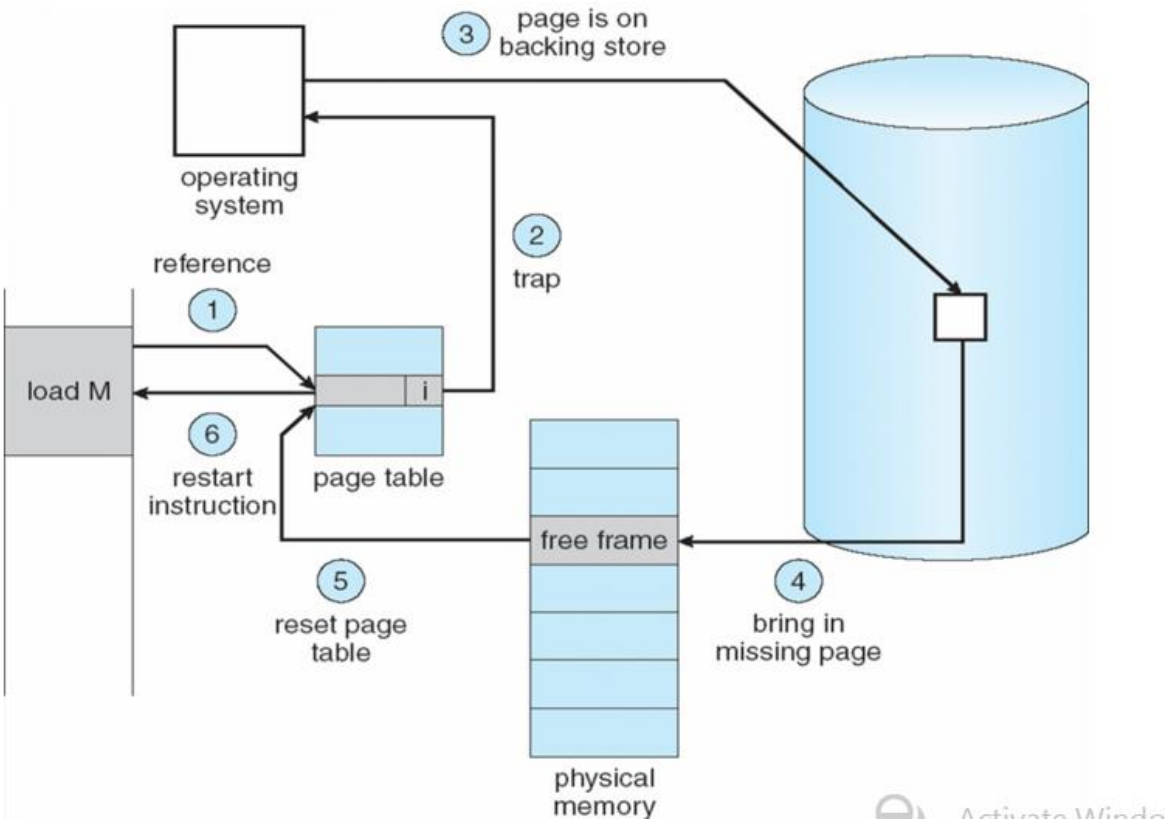
Segmentation

Basic Method of Segmentation

- This is a memory-management scheme that supports user-view of memory (Figure 1).
- A logical-address space is a collection of segments.
- Each segment has a name and a length.
- The addresses specify both segment-name and offset within the segment.
- Normally, the user-program is compiled, and the compiler automatically constructssegments reflecting the input program.
- For ex: The code, Global variables, The heap, from which memory is allocated, Thestacks used by each thread, The standard C library

3b. Describe the steps involved in handling a page fault.

[5.0] 1 [2]



If a page is needed that was not originally loaded up, then a ***page fault trap*** is generated.

Steps in Handling a Page Fault

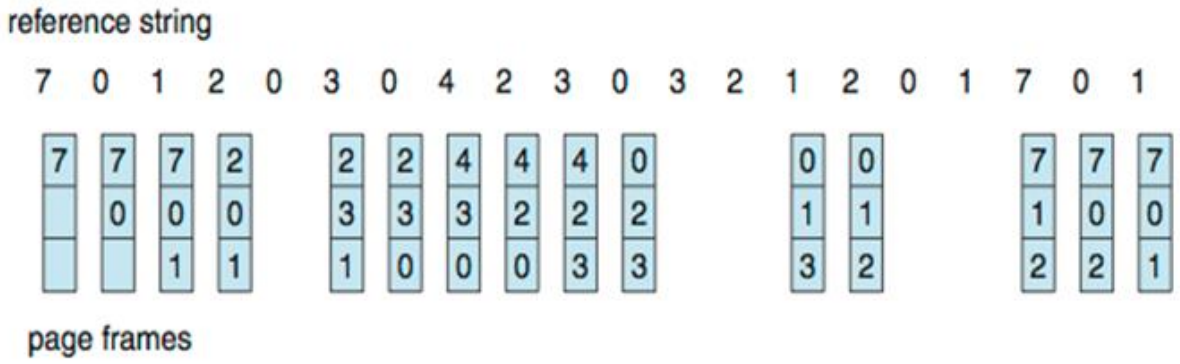
1. The memory address requested is first checked, to make sure it was a valid memoryrequest.
2. If the reference is to an invalid page, the process is terminated. Otherwise, if the page isnot present in memory, it must be paged in.
3. A free frame is located, possibly from a free-frame list.
4. A disk operation is scheduled to bring in the necessary page from disk.
5. After the page is loaded to memory, the process's page table is updated with the new frame number, and the invalid bit is changed to indicate that this is now a valid page reference.
6. The instruction that caused the page fault must now be restarted from the beginning.

4a. Consider the following page reference string.

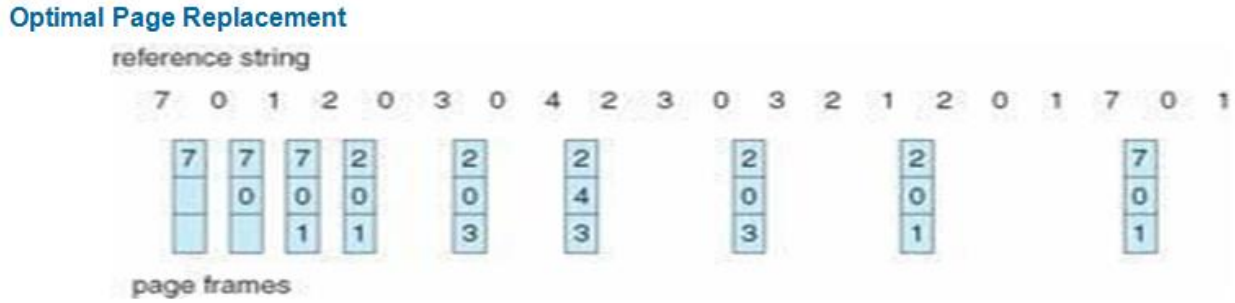
7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1.

How many page faults would occur for LRU, FIFO and optimal page replacement algorithm assuming 3 frames.

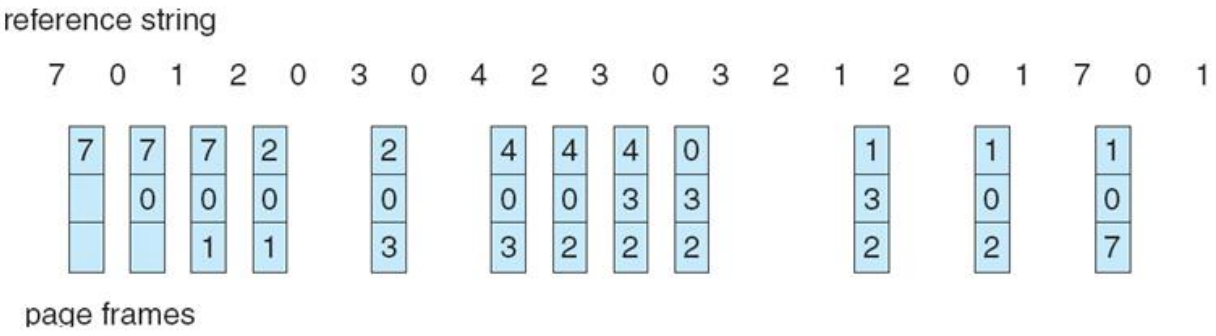
FIFO



Page Hit= 5, Page Miss= 15



Page Hit= 11, Page Miss= 09



Page Hit=8 , Page Miss= 12

[6.0] 1 [3]

4b. Given the memory partitions of 100k,500k,200k,300k, 600k apply first fit , best fit and worst fit to place 212k, 417k,112k and 426k

Let p1, p2, p3 & p4 are the names of the processes

a. First-fit:

- **P1**>>> 100, **500**, 200, 300, 600
- **P2**>>> 100, 288, 200, 300, **600**
- **P3**>>> 100, **288**, 200, 300, 183
- 100, 116, 200, 300, 183 <<<<< **final set of hole**
- P4 (426K) must wait

b. Best-fit:

- **P1**>>> 100, 500, 200, **300**, 600
- **P2**>>> 100, **500**, 200, 88, 600
- **P3**>>> 100, 83, **200**, 88, 600
- **P4**>>> 100, 83, 88, 88, **600**
- 100, 83, 88, 88, 174 <<<<< **final set of hole**

c. Worst-fit:

- **P1**>>> 100, 500, 200, 300, **600**
- **P2**>>> 100, **500**, 200, 300, 388
- **P3**>>> 100, 83, 200, 300, **388**
- 100, 83, 200, 300, 276 <<<<< **final set of hole**
- P4 (426K) must wait

In this example, Best-fit turns out to be the best **because there is no wait processes.**

[4.0] 1 [3]

5a. What are TLB? With a neat diagram explain TLB in detail with a simple paging system .

[5.0] 1 [2]

Translation Look aside Buffer

- A special, small, fast lookup hardware cache, called a translation look-aside buffer(TLB).
- Each entry in the TLB consists of two parts: a key (or tag) and a value.
- When the associative memory is presented with an item, the item is compared with all keys simultaneously. If the item is found, the corresponding value field is returned. The search is fast; the hardware, however, is expensive. Typically, the number of entries in a TLB is small, often numbering between 64 and 1,024.
- The TLB contains only a few of the page-table entries.

Working:

- When a logical-address is generated by the CPU, its page-number is

presented to theTLB.

- If the page-number is found (TLB hit), its frame-number is immediately available and used to access memory
- If page-number is not in TLB (TLB miss), a memory-reference to page table must be made. The obtained frame-number can be used to access memory (Figure 1)

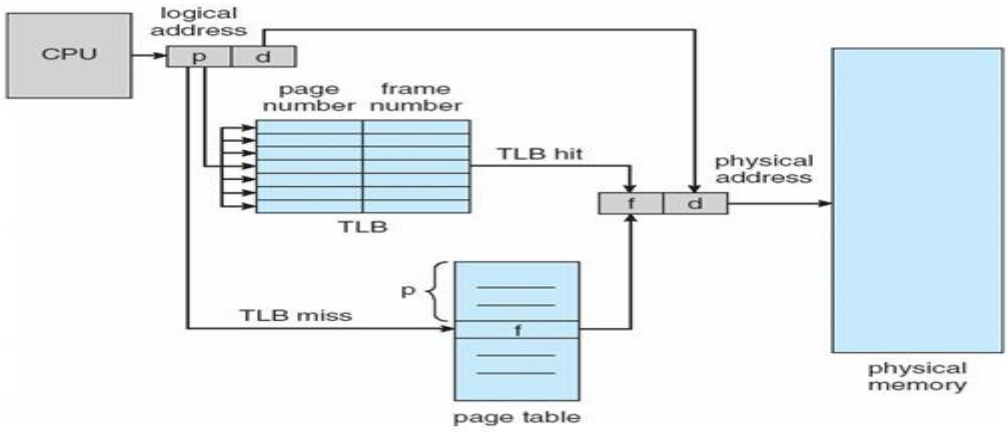


Figure 1: Paging hardware with TLB

Figure 1: Paging hardware with TLB

- In addition, we add the page-number and frame-number to the TLB, so that they will befound quickly on the next reference.
- If the TLB is already full of entries, the OS must select one for replacement.
- Percentage of times that a particular page-number is found in the TLB is called hit ratio.

Advantage: Search operation is fast.
Disadvantage: Hardware is expensive.

- Some TLBs have wired down entries that can't be removed.
- Some TLBs store ASID (address-space identifier) in each entry of the TLB that uniquelyidentify each process and provide address space protection for that process.

5b. Explain the structure of page table.

[5.0] 1 [2]

The most common techniques for structuring the page table:

1. Hierarchical Paging
2. Hashed Page-tables
3. Inverted Page-tables

1. Hierarchical Paging

- Problem: Most computers support a large logical-address space (232 to 264). In thesesystems, the page-table itself becomes excessively large.
- Solution: Divide the page-table into smaller pieces.

Two Level Paging Algorithm:

- The page-table itself is also paged.
- This is also known as a forward-mapped page-table because address translation worksfrom the outer page-table inwards.

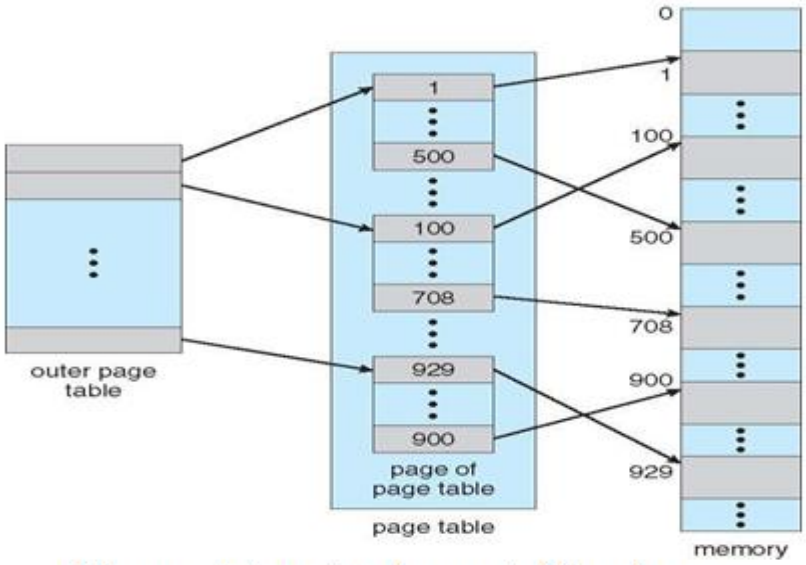


Figure: A two-level page-table scheme

2. Hashed Page Tables

- This approach is used for handling address spaces larger than 32 bits.
- The hash-value is the virtual page-number.
- Each entry in the hash-table contains a linked-list of elements that hash to the samelocation (to handle collisions).
- Each element consists of 3 fields:
    1. Virtual page-number

2. Value of the mapped page-frame and
3. Pointer to the next element in the linked-list.

The algorithm works as follows:

- The virtual page-number is hashed into the hash-table.
- The virtual page-number is compared with the first element in the linked-list.
- If there is a match, the corresponding page-frame (field 2) is used to form the desiredphysical-address.
- If there is no match, subsequent entries in the linked-list are searched for a matchingvirtual page-number.

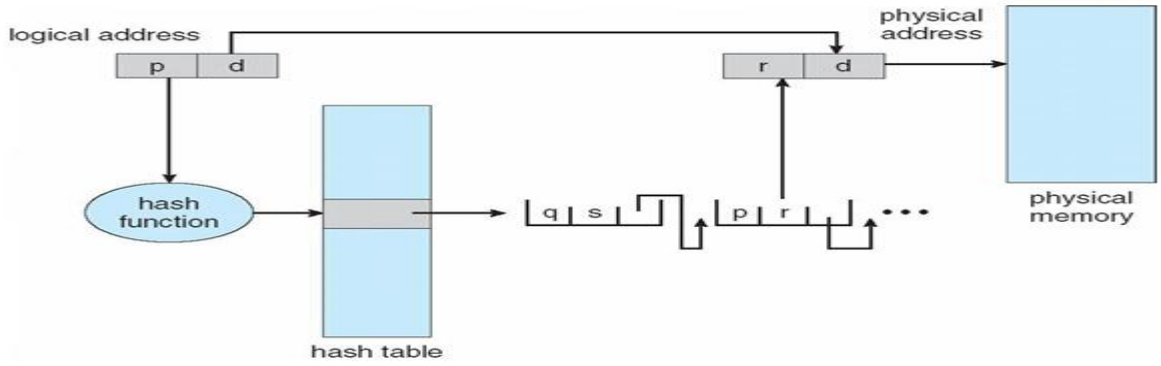
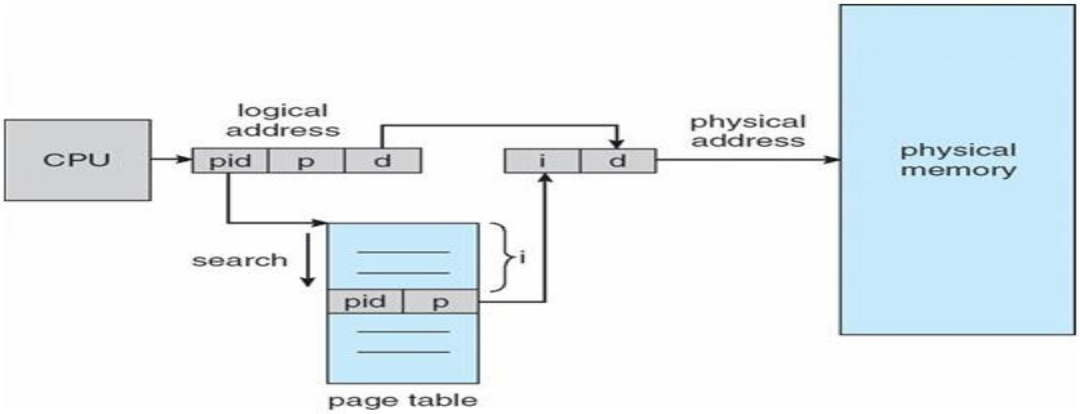


Figure: Hashed page-table

3. Inverted Page Tables

- Has one entry for each real page of memory.
- Each entry consists of virtual-address of the page stored in that real memory-locationand information about the process that owns the page.
- Each virtual-address consists of a triplet <process-id, page-number, offset>.
- Each inverted page-table entry is a pair <process-id, page-number>



- Figure: Inverted page-table

The algorithm works as follows:

1. When a memory-reference occurs, part of the virtual-address, consisting of <process-id,page-number>, is presented to the memory subsystem.
2. The inverted page-table is then searched for a match.
3. If a match is found, at entry i-then the physical-address <i, offset> is generated.
   If no match is found, then an illegal address access has been attempted

6. Discuss the various approaches used for deadlock recovery

[10.0] 1 [2]

RECOVERY FROM DEADLOCK

The system recovers from the deadlock automatically. There are two options for breaking a deadlock one is simply to abort one or more processes to break the circular wait. The other is topreempt some resources from one or more of the deadlocked processes.

Process Termination

To eliminate deadlocks by aborting a process, use one of two methods. In both methods, thesystem reclaims all resources allocated to the terminated processes.

1. **Abort all deadlocked processes:** This method clearly will break the deadlock cycle, butat great expense; the deadlocked processes may have computed for a long time, and the results of these partial computations must be discarded and probably will have to be recomputed later.
2. **Abort one process at a time until the deadlock cycle is eliminated:** This method incurs considerable overhead, since after each process is aborted, a deadlock-detectionalgorithm must be invoked to determine whether any processes are still deadlocked.

If the partial termination method is used, then we must determine which deadlocked process (orprocesses) should be terminated. Many factors may affect which process is chosen, including:

1. What the priority of the process is
2. How long the process has computed and how much longer the process will computebefore completing its designated task
3. How many and what types of resources the process has used.
4. How many more resources the process needs in order to complete

5. How many processes will need to be terminated?
6. Whether the process is interactive or batch

Resource Preemption

To eliminate deadlocks using resource preemption, we successively preempt some resourcesfrom processes and give these resources to other processes until the deadlock cycle is broken.
If preemption is required to deal with deadlocks, then three issues need to be addressed:

1. **Selecting a victim.** Which resources and which processes are to be preempted? As in process termination, we must determine the order of preemption to minimize cost. Cost factors may include such parameters as the number of resources a deadlocked process is holding and the amount of time the process has thus far consumed during its execution.
2. **Rollback.** If we preempt a resource from a process, what should be done with that process? Clearly, it cannot continue with its normal execution; it is missing some needed resource. We must roll back the process to some safe state and restart it from that state. Since it is difficult to determine what a safe state is, the simplest solution is a total rollback: abort the process and then restart it.
3. **Starvation.** How do we ensure that starvation will not occur? That is, how can weguarantee that resources will not always be preempted from the same process?

7. Consider the following segment table

SEGMENT	BASE	LENGTH
0	219	600
1	2300	14
2		
90	100	
3	1327	580
4	1952	96

What are the physical addresses for the following logical address.

a)0,430 b)1,10 c) 2,500 d) 3,400 e) 4,112

[10.0] 1 [3]

3. Consider the following segment table:

Segment	Base	Length
0	219	600
1	2300	14
2	90	100
3	1327	580
4	1952	96

What are the physical addresses for the following logical addresses?

a. 0,430

b. 1,10

c. 2,500

d. 3,400

e. 4,112

Solution:

a. 219 + 430 = 649

b. 2300 + 10 = 2310

c. illegal reference, trap to operating system

d. 1327 + 400 = 1727

e. illegal reference, trap to operating system