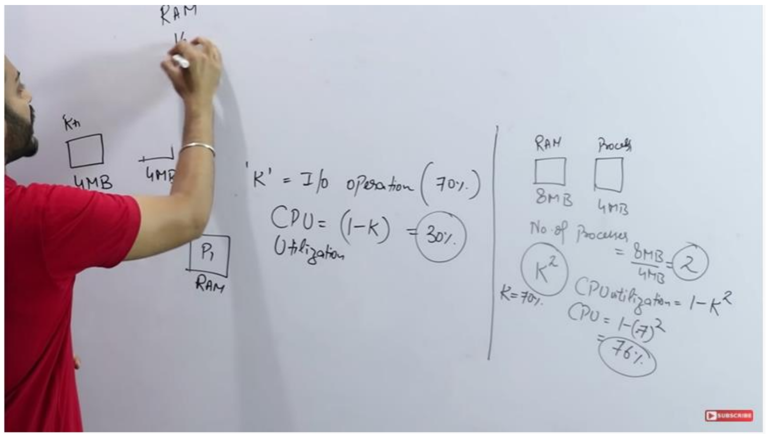
MEMORY MANAGENEMT

The method on managing primary memory, goal is Efficient utilization of memory.

Programs stay idle in secondary memory; they are brought to Primary Memory (RAM) to be executed.



Degree of Multiprogramming is directly proportional to the size of RAM.

SOME POINTS ABOUT MEMORY MANAGEMENT

1. Main memory and register are only storage that a CPU can access directly.
2. Cache sits between main memory and register.

BASE AND LIMIT REGISTERS

A pair of base and limit registers define the logical address space.

The base registers indicate where the page table starts in memory (Physical or Logical addresses) and the limit register indicates the side of the table.

ADDRESS BINDING

Address binding is the process of mapping from one address space to another address space. Logical address is address generated by CPU during execution whereas Physical Address refers to location in memory unit (the one that is loaded into memory). Note that user deals with only logical address(Virtual address). The logical address undergoes translation by the MMU or address translation unit. The output of this process is the appropriate physical address or the location of code/data in RAM.

An address binding can be done in three different ways:

Compile Time – If you know that during compile time where process will reside in memory then absolute address is generated i.e physical address is embedded to the executable of the program during compilation. Loading the executable as a process in memory is amazingly fast. But if the generated address space is preoccupied by other process, then the program crashes and it becomes necessary to recompile the program to change the address space.

Load time – If it is not known at the compile time where process will reside then relocatable address will be generated. Loader translates the relocatable address to absolute address. The base address of the process in main memory is added to all logical addresses by the loader to generate absolute address. In this if the base address of the process changes then we need to reload the process again.

Execution time- The instructions are in memory and are being processed by the CPU. Additional memory may be allocated and/or deallocated at this time. This is used if process can be moved from one memory to another during execution (dynamic linking-Linking that is done during load or run time). e.g – Compaction.

DYNAMIC LINKING AND STATIC LINKING

STATIC LINKING

When we click the .exe (executable) file of the program and it starts running, all the necessary contents of the binary file have been loaded into the process's virtual address space. However, most programs also need to run functions from the system libraries, and these library functions also need to be loaded.

In the simplest case, the necessary library functions are embedded directly in the program's executable binary file. Such a program is statically linked to its libraries, and statically linked executable codes can commence running as soon as they are loaded.

DYNAMIC LINKING

Every dynamically linked program contains a small, statically linked function that is called when the program starts. This static function only maps the link library into memory and runs the code that the function contains. The link library determines what are all the dynamic libraries which the program requires along with the names of the variables and functions needed from those libraries by reading the information contained in sections of the library.

After which it maps the libraries into the middle of virtual memory and resolves the references to the symbols contained in those libraries. We don't know where in the memory these shared libraries are actually mapped: They are compiled into position-independent code (PIC), that can run at any address in memory.

MEMORY MANAGEMENT TECHNIQUES

CONTIGUOUS

Fixed Partition (Static):

Fixed sized memory allocated to processes in a contiguous manner.

- No of partitions are fixed.
- Size of partitions may or may not be same.
- Panning (Half process in a different block, half in different) is not allowed.

OS	4 MB	8 MB	8 MB	16 MB

OS	8 MB	8 MB	8 MB	8 MB

Problems:

1. Internal Fragmentation: When the size of memory block is more than the size of process. The remaining memory of that block is wasted. Process 2 MB, Memory Block 4 MB.
2. Limit in process size.
3. Degree of Multi programming is limited (No more than 4 processes can be accommodated in the above memory blocks).

Variable Partition (Dynamic):

Differed sized memory allocated to processes in a contiguous manner. Size is allocated on run time.

Advantages:

1. No internal fragmentation.
2. No limitation on number of processes or multiprogramming.
3. No limitation on process size.

P1 = 2 MB, P2 = 4 MB, P3 = 3 MB, P4 = 4 MB, P5 = 2 MB

2 MB	4 MB	3 MB	4 MB	2 MB

P2 and P4 are deallocated, P6 = 8 MB arrives, even though collective 8 MB is available it is not contiguous, it is a problem of external fragmentation, hence memory cannot be allocated to P6.

2 MB	HOLE	3 MB	HOLE	2 MB

ALLOCATION METHODS

FIRST FIT

Allocate first hole that is big enough.

P10	25KB	P11	40K	P12	100K	P12	20K	P4	10K
	P1		P2						
	10K		22K						

P1 = 15K, P2 = 18K → Waste: 32K

NEXT FIT

Same as first fit but start search from last allocated hole.

BEST FIT

Allocate the smallest hole that is big enough.

P10	25KB	P11	40K	P12	100K	P12	20K	P4	10K
	P2						P1		
	7K						5K		

P1 = 15K, P2 = 18K → Waste: 12K

WORST FIT

Allocate the largest hole.

P10	25KB	P11	40K	P12	100K	P12	20K	P4	10K
			P2		P1				
			22K		85K				

P1 = 15K, P2 = 18K → Waste: 107K

NON – CONTIGUOUS

Here, a process can span at different memory locations.

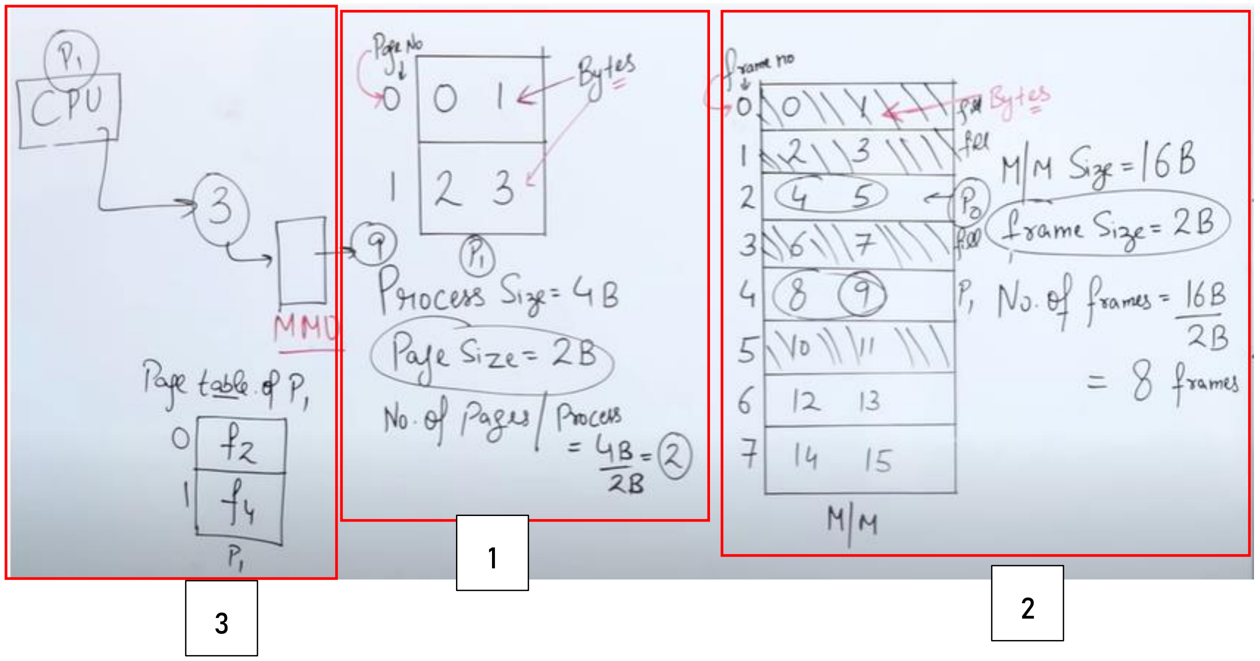
1. Paging
   Processes are divided into pages, and the memory blocks are divided into frames.
   Page size = Frame size.
   P1 = 3kb, P2 = 2kb, P3 = 1kb, P4 = 2kb.

1 kb	1 kb	1 kb	1 kb	1 kb	1 kb	1 kb	1 kb
P1	P1	P1	P2	P3	P2	P4	P4

2. Multi - Level Paging
3. Inverted Paging
4. Segmentation
5. Segmented Paging

Processes are put in different partitions of memory in RAM in non – contiguous memory.

WHAT IS PAGING



CPU Always working on Logical Address, and logical address is made up of two things

1. Page Number
2. Page Offset (Size of the Page)

28/04/2021 VIDEO (GCR)

VIRTUAL MEMORY

Virtual memory provides an illusion to the user that a program whose size is larger than the main memory can also be executed on the system.

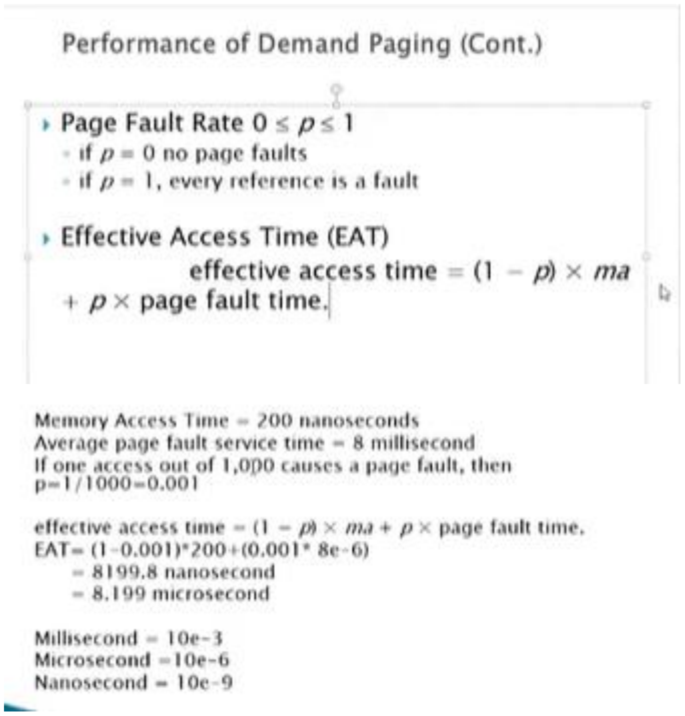
The program is stored of the physical memory and a few required pages are mounted on the main memory.

This work is done by swap/roll in and swap/roll out process, also known as page replacement.

DEMAND PAGING

Load a page only into the memory when needed.

1. Lesser IO needed, no wastage of IO.
2. Lesser memory needed.
3. Faster response.



Performance of Demand Paging (Cont.)

- Page Fault Rate $0 \le p \le 1$
  - if $p = 0$ no page faults
  - if $p = 1$, every reference is a fault
- Effective Access Time (EAT)

  effective access time $= (1 - p) \times ma + p \times$ page fault time.

Memory Access Time = 200 nanoseconds
Average page fault service time = 8 millisecond
If one access out of 1,000 causes a page fault, then
p=1/1000=0.001

effective access time $= (1 - p) \times ma + p \times$ page fault time.
EAT= (1-0.001)*200+(0.001* 8e-6)
= 8199.8 nanosecond
= 8.199 microsecond

Millisecond = 10e-3
Microsecond =10e-6
Nanosecond = 10e-9

TLB

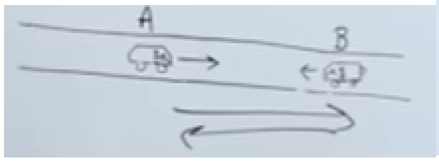
VALID – INVALID BIT

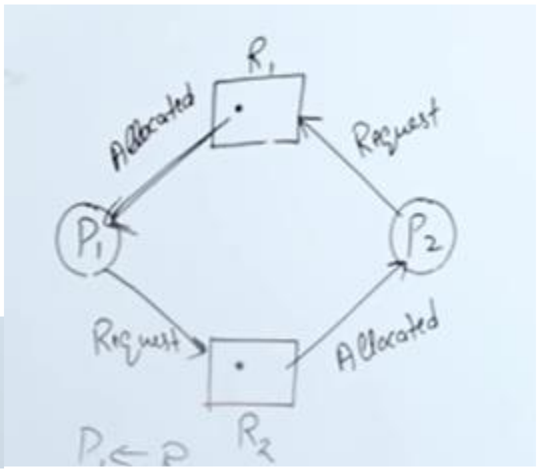
With each page table entry, there is a Valid and Invalid bit associated, which tells the system if the page exists in the table or not.

An Invalid bit created page fault.

DEADLOCK

If two or more process are waiting for some event to happen, but that process never happens, then we say that these processes are in deadlock situation.





DEADLOCK CONDITIONS

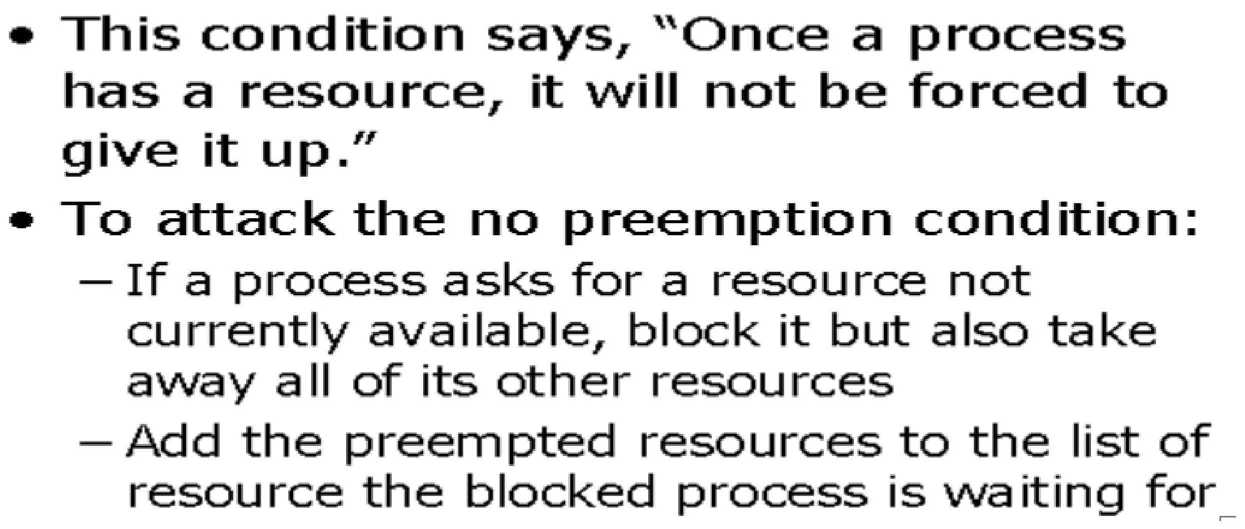
1. MUTUAL EXCLUSION

- This condition says, "There exist resources in the system that can be used by only one process at a time."
- Examples include printer, write access to a file or record, entry into a section of code

2. NO PRE-EMPTION

- This condition says, "Once a process has a resource, it will not be forced to give it up."
- To attack the no preemption condition:
  - If a process asks for a resource not currently available, block it but also take away all of its other resources
  - Add the preempted resources to the list of resource the blocked process is waiting for

3. HOLD AND WAIT



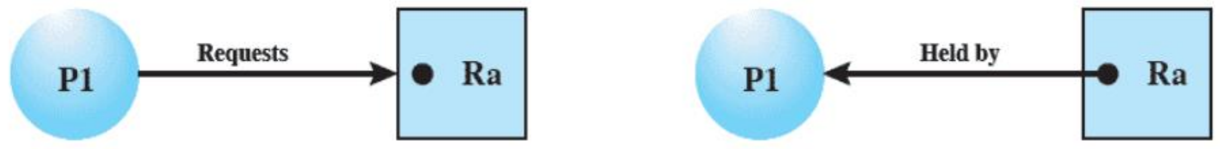
- This condition says, "Once a process has a resource, it will not be forced to give it up."
- To attack the no preemption condition:
  - If a process asks for a resource not currently available, block it but also take away all of its other resources
  - Add the preempted resources to the list of resource the blocked process is waiting for

4. CIRCULAR WAIT

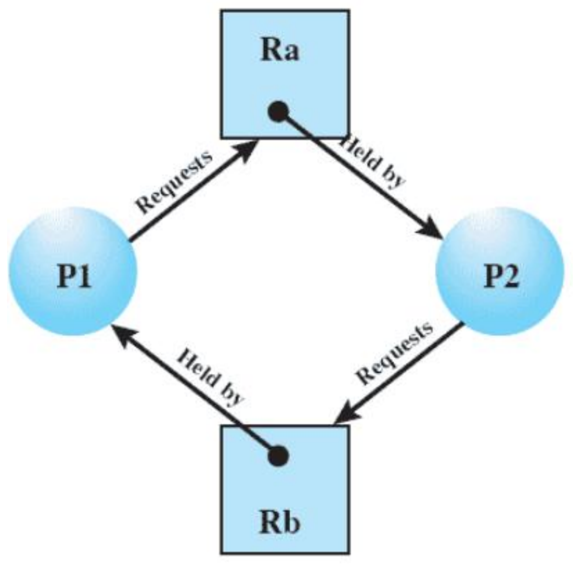
The diagram above explains circular wait visually.

RESOURCE ALLOCATION GRAPH

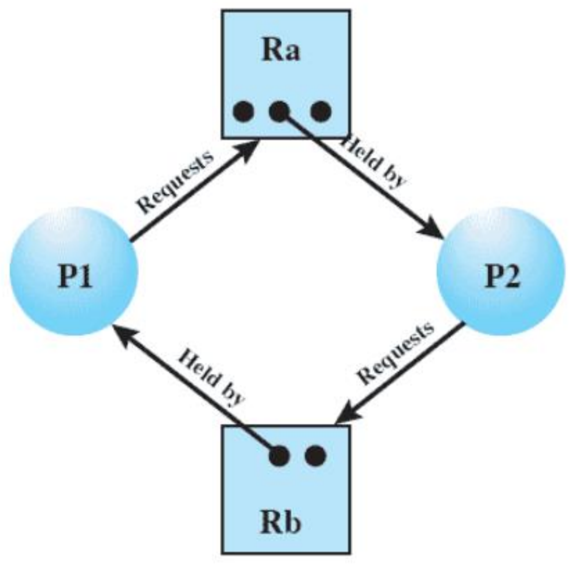
A directed graph that depicts the state of the system of resources and processes.



(a) Resouce is requested (b) Resource is held



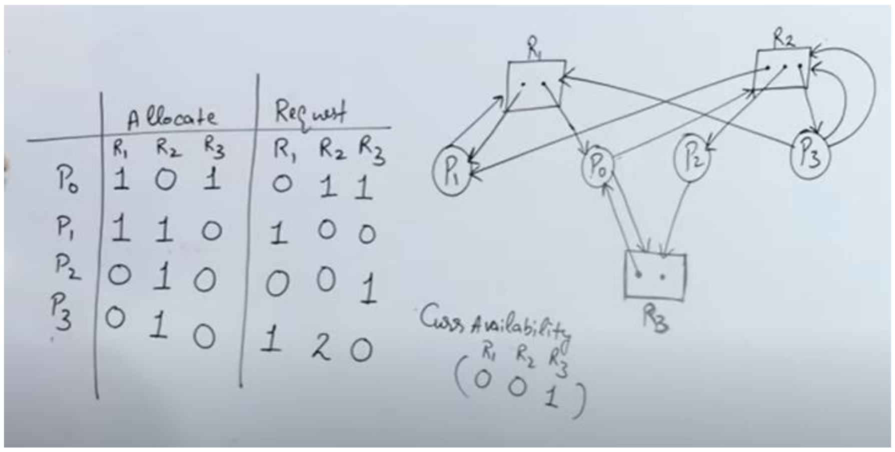
(c) Circular wait



(d) No deadlock

In a single instance RAG with a circular wait, there is always a deadlock.

RAG NUMERICAL



	Allocate			Request		
	R_1	R_2	R_3	R_1	R_2	R_3
P_0	1	0	1	0	1	1
P_1	1	1	0	1	0	0
P_2	0	1	0	0	0	1
P_3	0	1	0	1	2	0

Curr Availability

$$\begin{array}{ccc} R_1 & R_2 & R_3 \\ 0 & 0 & 1 \end{array}$$

APPROACHES TO DEADLOCK AVOIDANCE

PROCESS INITIATION DENIAL

A process is only started if the maximum claim of all current processes plus those of the new process can be met.

RESOURCE ALLOCATION DENIAL

- Referred to as the banker's algorithm
  - A strategy of resource allocation denial
- Consider a system with fixed number of resources
  - ***State*** of the system is the current allocation of resources to process
  - ***Safe state*** is where there is at least one sequence that does not result in deadlock
  - ***Unsafe state*** is a state that is not safe