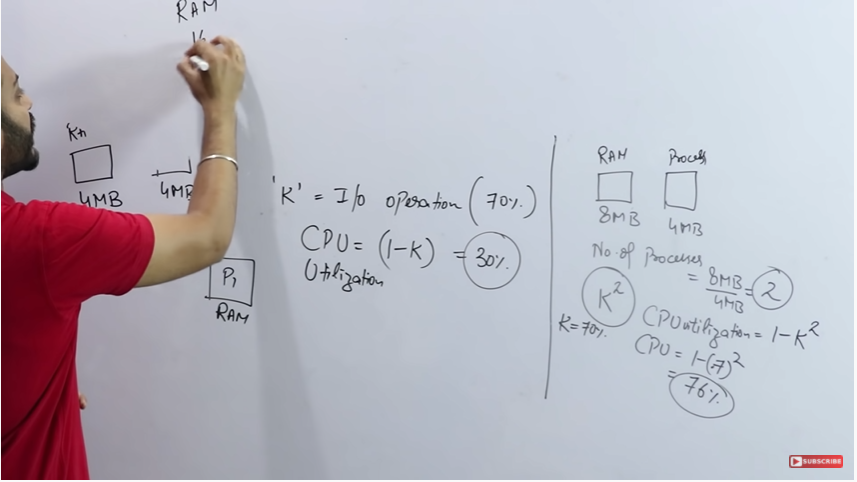
# Memory Managenemt

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.

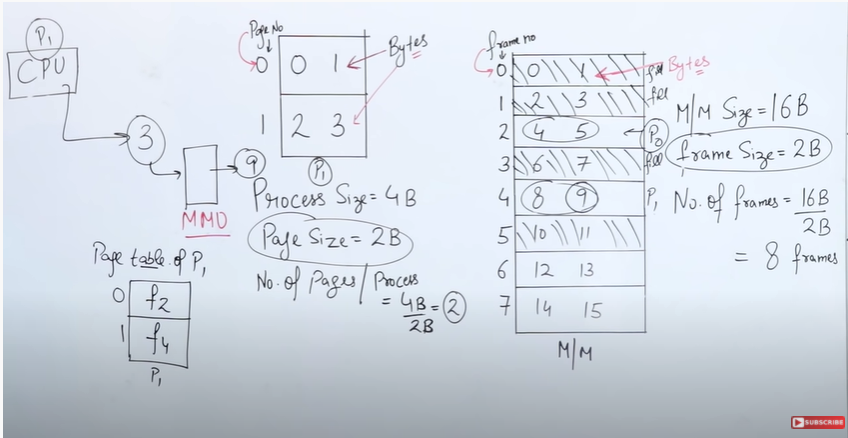
P1 = 3kb, P2 = 2kb, P3 = 1kb, P4 = 2kb.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 kb | 1 kb | 1 kb | 1 kb | 1 kb | 1 kb | 1 kb | 1 kb |
| P1 | P1 | P1 | P2 | P3 | P2 | P4 | P4 |

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

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

# What is Paging



3

2

1

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.

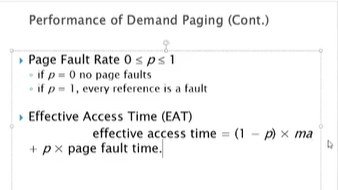
The program is stored of the physical memory and a few required pages are mounted on the main memory.

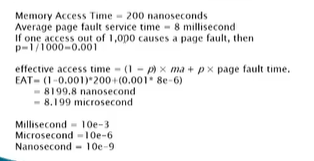
This work is done by swap/roll in and swap/roll out process, also known as page replacement.

## 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.





## TLB

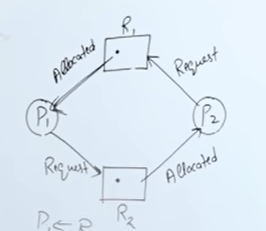
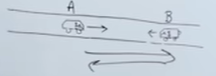
## Valid – InValid Bit

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.

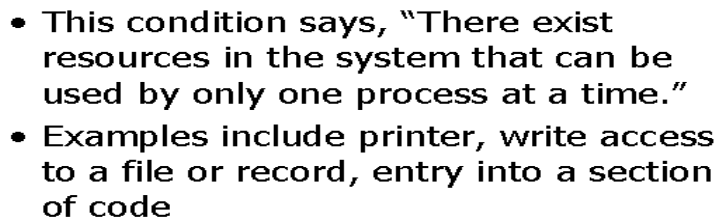
# DeadLock

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.

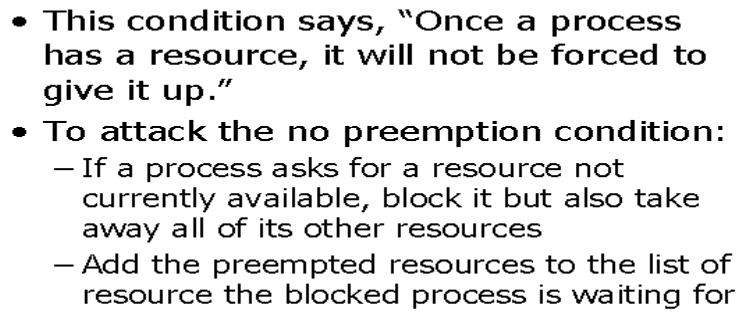


## DeadLock Conditions

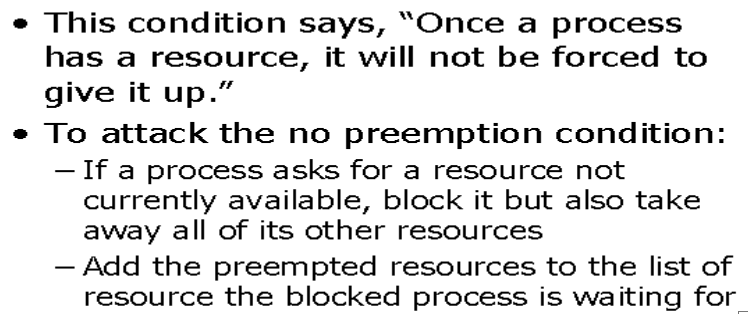
### Mutual Exclusion



### No Pre-Emption



### Hold and Wait

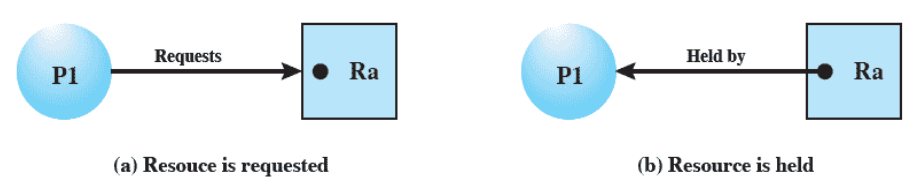


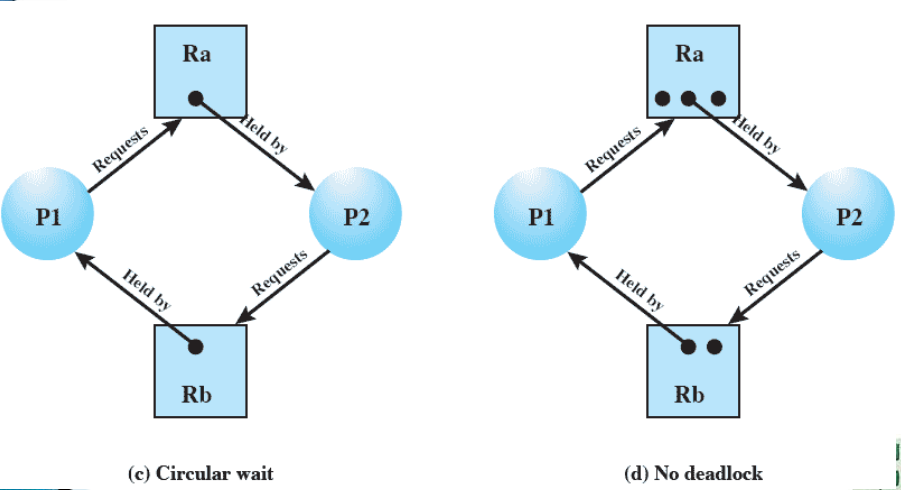
### Circular Wait

The diagram above explains circular wait visually.

## Resource Allocation Graph

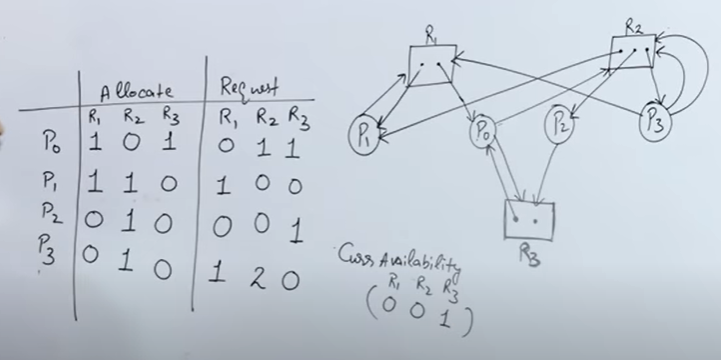
A directed graph that depicts the state of the system of resources and processes.





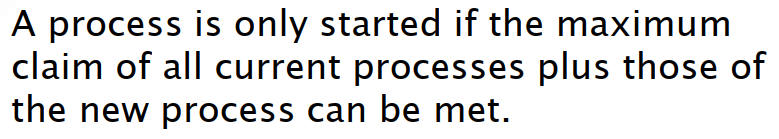
In a single instance RAG with a circular wait, there is always a deadlock.

### RAG Numerical



## Approaches to Deadlock Avoidance

### Process Initiation Denial



### Resource Allocation Denial

