Operating Systems All PPT

Topic- 1

Synchronization Problem

Presentation By :
Saurabh
19CSXXX CSE2

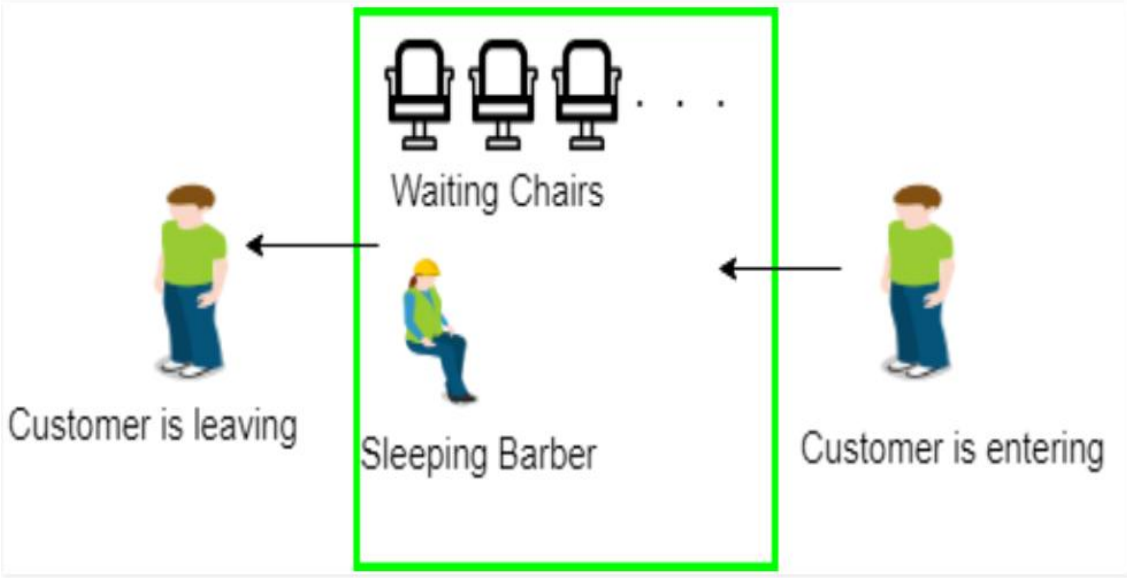
Process Synchronization is the task of coordinating the execution of processes in a way that no two processes can have access to the same shared data and resources.
Process is categorized into two types on the basis of synchronization:

1. Independent Process
2. Cooperative Process

Example:

In this topic, I am going solve Process Synchronization Problem.

- Sleeping Barber problem in Process Synchronization



- There is a barber shop which has one barber, one barber's chair, and n chairs for waiting for customers.

1. If there is no customer, then the barber sleeps in his own chair.
2. When a customer arrives, he has to wake up the barber.
3. If there are many customers and the barber is cutting a customer's hair, then the remaining customers have to wait.
4. if there are no empty chairs in the waiting room so customer leave or may go to others barber's shop.

Algorithm

19CS1100, Saurabh Kishor

```
Semaphore Customers = 0;
Semaphore Barber = 0;
Mutex Seats = 1;
int FreeSeats = N;

Barber {
        while(true) {

        down(Customers);
        down(Seats);

        FreeSeats++;

        up(Barber);
        up(Seats); }
}
```

```
Customer {
        while(true) {

        down(Seats);
        if(FreeSeats > 0) {
        FreeSeats--;
           up(Customers);
        up(Seats);
        down(Barber);
        }
        else {

        up(Seats);

        }
        }
}
```

Operating Systems

Deadlock Detection using resource allocation graph

-: Done By

Saurabh Kishor
19CSXXX CSE2

Deadlock Detection using resource allocation graph

Deadlock Detection-

Way to Detection, deadlock condition of processes, using Resource Allocation Graph. RAG help us to detected whether system is in a Deadlock state or not.

Deadlock: A situation occurs in operating systems when there are two or more processes that hold some resources and wait for resources held by other.

Detection means it helps us to find if there is any possibility of deadlock occurrence between processes in the future.

There are two thing :

1. vertex (Process and Resource)
2. Edge (Assign Edge and Request Edge)

Introduction

Process vertex :

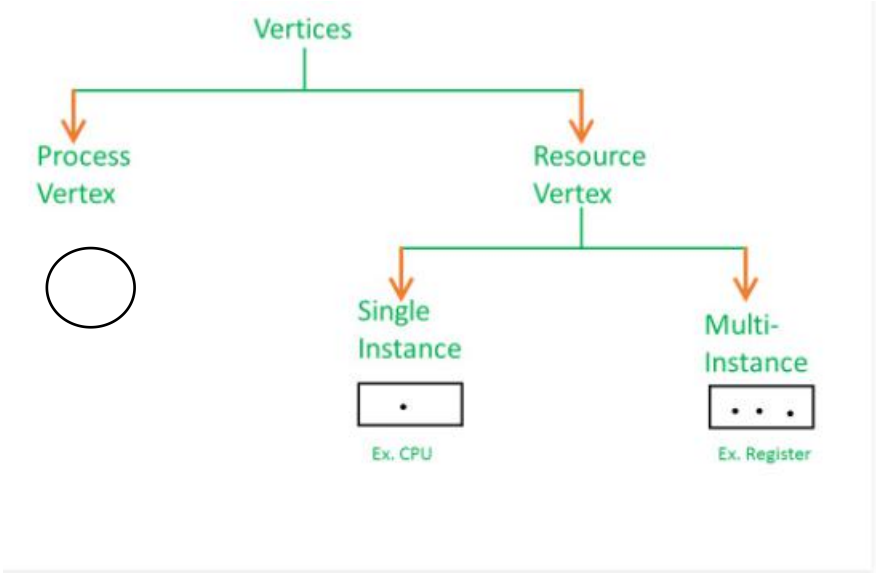
- Every process will be represented as a process vertex.
- Generally, the process will be represented with a circle.

2. **Resource vertex :** Every resource will be represented as a resource vertex.

It is also two type :

- Single instance type resource – It represents as a box, inside the box, there will be one dot.So the number of dots indicate how many instances are present of each resource type.

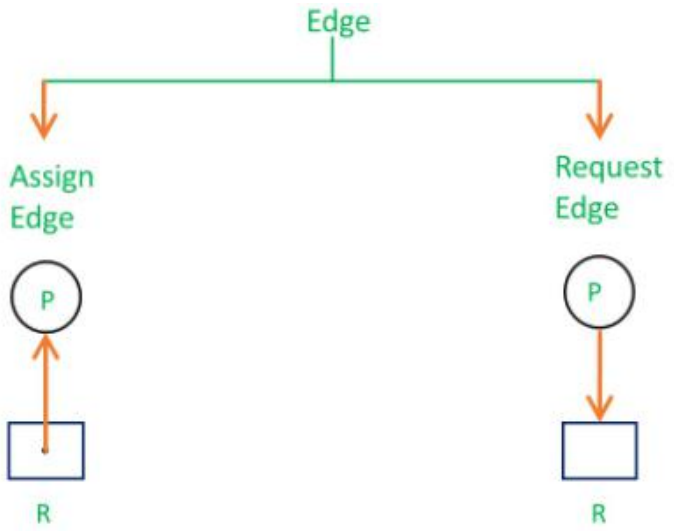
- Multi-resource instance type resource – It also represents as a box, inside the box, there will be many dots present.



About Edges

Assign Edge: When resource is Assign to a process is called assign edge.

Request Edge – When Process need a Resource, then it request for request is called request edge.



Deadlock Detection (By RAG)

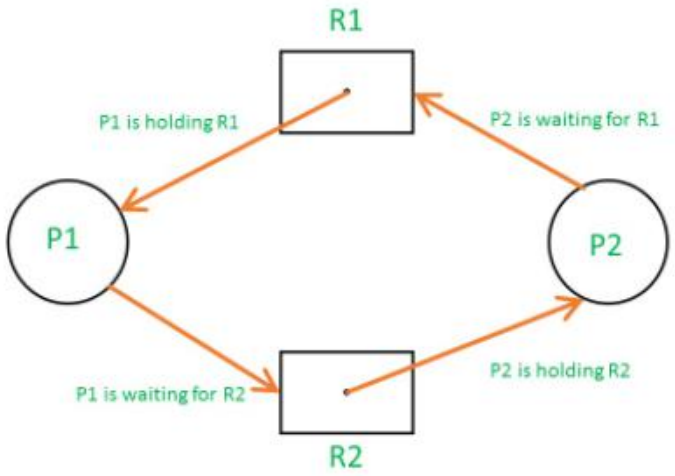
Case:1

All the resources have single instance:

So for detect deadlock using Resource Allocation Graph, when all the resources have single instance.
check these condition:

1.If a cycle is being formed, then system is in a deadlock state.

2.If no cycle is being formed, then system is not in a deadlock state.



SINGLE INSTANCE RESOURCE TYPE WITH DEADLOCK

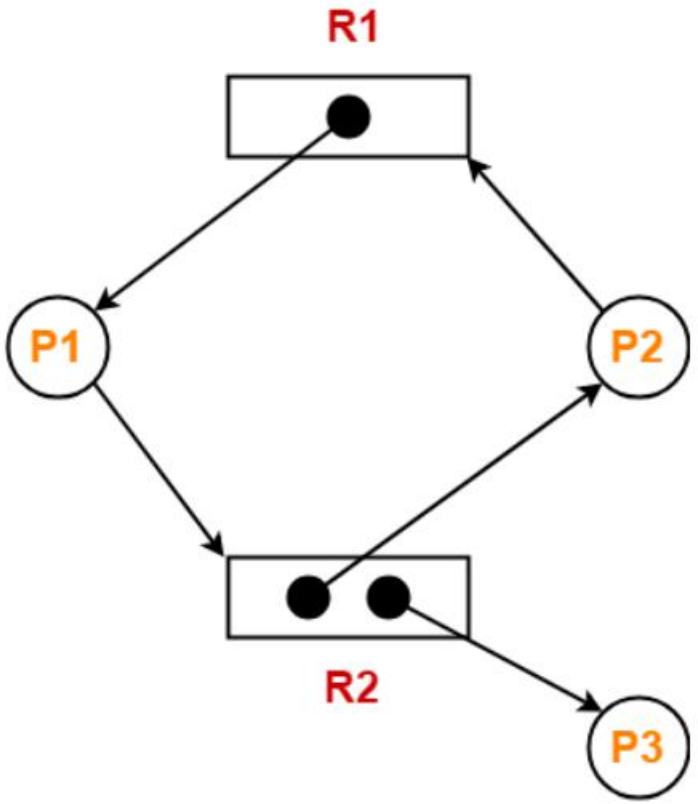
Deadlock detection using RAG in Multiple instance resource

19CS1100, Saurabh Kishor

Case:2

All the resources have NOT single instance, may be two or more than two instance:

1.If a cycle is being formed, then system may be in a deadlock state or may not be. Because, Presence of a cycle is a necessary but not a sufficient condition for the occurrence of deadlock.



2.If no cycle is being formed, then system is not in a deadlock state.

How to detect deadlock by RAG ?

From the above example, it is not possible to say the RAG is in a safe state or in an unsafe state.So to see the state of this RAG, let's construct the allocation matrix and request matrix.

Allocation matrix :

For allocation matrix, we find out, which process it is allocated. R1 is allocated to P1, therefore write 1 in allocation matrix and similarly, R2 is allocated to P2 as well as P3 and for the remaining element just write 0.

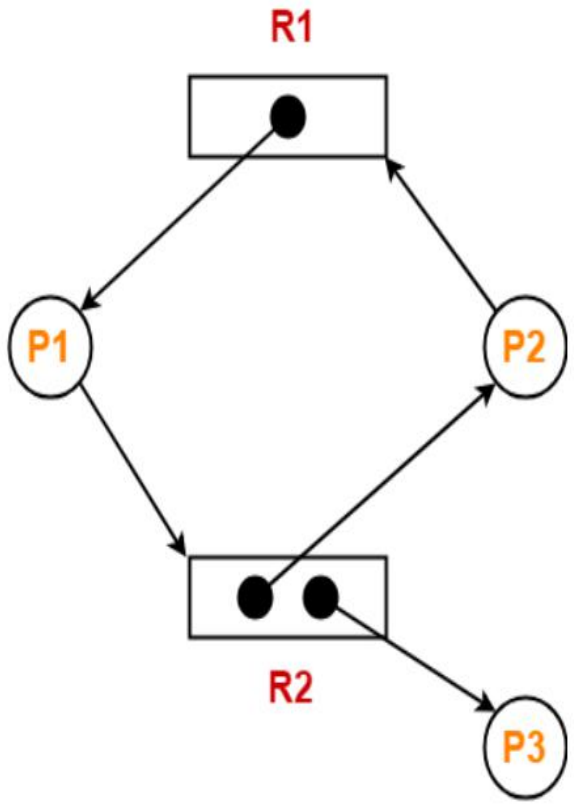
Process	Allocation		Request	
	Resource		Resource	
	R1	R2	R1	R2
P1	1	0	0	1
P2	0	1	1	0
P3	0	1	0	0

Request matrix :

From request matrix, we find out
P1 is requesting resource R2, so write 1 in the matrix and similarly, P2 requesting R1 and for the remaining element write 0.
So now available resource is = (0, 0).

Step-01:

Since process P3 does not need any resource, so it executes.
After execution, process P3 release its resources.
Then,
Available
= [0 0] + [0 1]
= [0 1]



Step-02:
With the instances available currently, only the requirement of the process P1 can be satisfied. So, process P1 is allocated the requested resources.
It completes its execution and then free up the instances of resources held by it.
Then-
Available
= [0 1] + [1 0]
= [1 1]
Step-03:
With the instances available currently, the requirement of the process P2 can be satisfied.
So, process P2 is allocated the requested resources.
It completes its execution and then free up the instances of resources held by it.

Then-
Available
= [1 1] + [0 1]
= [1 2]
Thus, There exists a safe sequence P3, P1, P2 in which all the processes can be executed.

Operating Systems

Banker's algorithm

-: Done By

Saurabh Kishor
19CSXXXX CSE2

Banker's algorithm

19CS1100, Saurabh Kishor

- suppose, we have 5 processes that are requesting some resources like here we have three namely A, B and C which may be physical similar to CPU, Memory and printer
- And these five processes are requesting for these resources of their individual needs And on this basis, I am applying the Banker's algorithm, we have have to find out whether deadlock occurs or not
- we assume that each resources have some instances like here A = 11, B = 6 and C = 8, in the sense 11 CPUs, 6 Memories and 8 Printers

- so our objective is to find , is there a deadlock exists or not? and also safe or unsafe. If it is safe then no deadlock. If unsafe then deadlock exists and if it a Safe then we find safe Sequence.

To check it is safe or not?

we find **Available = Total - Allocated**

Remining Need = max- Allocation

- In my example it a Safe Because deadlock has not occurred anywhere
- So the safe sequence is P2, P4, P5, P1, P3.
- It is not unique as the values could have swapped.

Banker's algorithm:

- Banker's algorithm is work as Deadlock Avoidance algorithm and
- Banker's Algorithm is also used for Deadlock Detection

Let suppose, we have 5 processes and 3 resources, and number of resources and also given amount of resource is allocated to respective process

Total Resource instances A = 11, B = 6, C = 8

Process	Allocation			Max			Available (Total -Allocated)			Remining Need (max- Allocation)		
	A cpu	B Memory	C printer	A	B	C	A	B	C	A	B	C
P1	1	0	2	8	3	5	2	3	4	7	3	3
P2	3	0	1	3	3	2						
P3	2	0	1	2	0	6						
P4	3	1	0	4	2	3						
P5	0	2	0	3	5	3						
	9	3	4									

Available = Total - Allocation

= (11,6,8) - (9,3,4)

= (2, 3, 4)

1. For P1:

Need = max - Allocation

= (8,3,5) - (1,0,2)

= 7, 3, 3

Need <= Available

7, 3, 3 <= 2, 3, 4 condition is **false.**

So, we examine another process, P2.

2. For P2:

Need = (3,3,2) - (3,0,1)

= 0, 3, 1

Need <= Available

0, 3, 1 <= 2, 3, 4 condition is **True**

so, P2 will go for execution.

New available = available + Allocation

= (2, 3, 4) + (3,0,1)

= (5,3,5)

3. For P3:

Need = (2,0,6) - (2,0,1)

= 0, 0, 5

Need <= Available

0, 0, 5 <= 5, 3, 5 condition is **True**

so, P3 will go for execution.

New available = available + Allocation

= (5, 3, 5) + (2,0,1) = (7,3,6)

4. For P4:

Need = (4,2,3) - (3,1,0)

= 1, 1, 3

Need <= Available

1, 1, 3 <= 7, 3, 6 condition is **True**

so, P4 will go for execution.

New available = available + Allocation

= (7, 3, 6) + (3, 1, 0)

= (10, 4, 6)

5. For P5:

Need = (3,5,3) - (0,2,0)

= (3,3,3)

Need <= Available

3, 3, 3 <= 10, 4, 6 condition is **True**

so, P5 will go for execution.

New available = available + Allocation

= (10, 4, 6) + (0, 2, 0)

= (10, 6, 6)

6. Again, go to P1

Need <= Available

7, 3, 3 <= 10, 6, 6 condition is **True.**

So, P1 will go for execution.

Remining Need (max- Allocation)		
A	B	C
7	3	3
0	3	1
0	0	5
1	1	3
3	3	3

Available (Total -Allocated)		
A	B	C
2	3	4
5	3	5
7	3	6
10	4	6
10	6	6

so, it a Safe Sequence because deadlock has not occurred anywhere.

Hence, Safe Sequence order: p2 -> p3 -> p4->p5 -> p1

Operating Systems

Logical to Physical Address Binding

-: Done By

Saurabh Kishor
19CSXXXX CSE2

Logical to Physical Address Binding

In this topic i am going to discuss about how to binding Virtual Addresses to Physical Addresses?

- Address binding is the process of mapping from one address space to another address space.

logical address: A **logical address** is generated by CPU while a program is running.

- it is also known as a virtual address because a logical address does not physically exists .
- Logiacl address space is the set of all logical addresses generated by the program.

Physical address : A physical address is generated by memory unit.

- Physical address space is the set of all physical addresses generated by the program

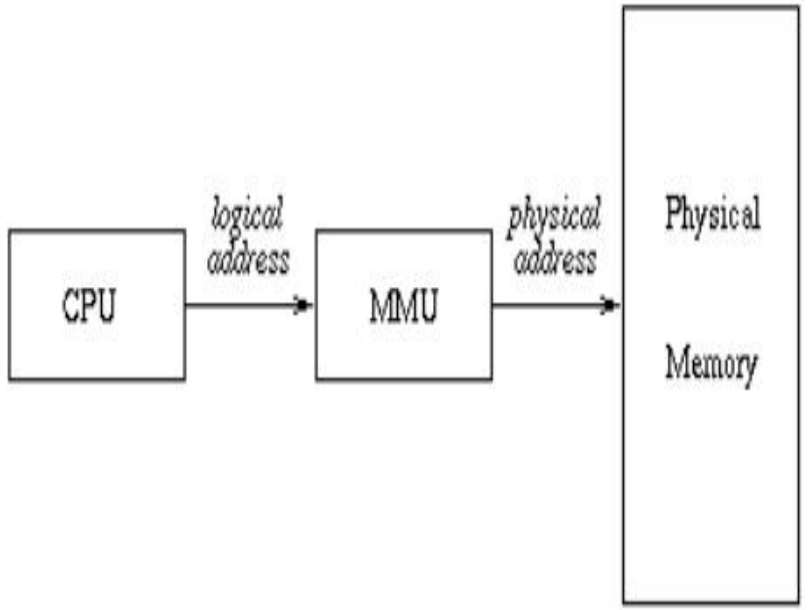
Logical and physical address are same at compile time And Differet at execution time.

Mapping Virtual Addresses to Physical Addresses

The run time mapping between virtual address and physical address is done by a hardware device known as **MMU (Memory Management Unit)**.

Mapping Virtual Addresses to Physical Addresses

- We can never directly deals with the physical address but we can determine the physical address by its the help of corresponding logical address.
- When user program generates the logical address and the program is running in this logical address space, but the program needs physical memory for its execution, therefore, the logical address must be mapped to the physical address by the MMU before the addresses are used.



Operating Systems

Contiguous memory allocation- First fit, Best fit and worst fit.

-: Done By

Saurabh Kishor
19CSXXX CSE2

Contiguous memory allocation- First fit, Best fit and worst fit.

Contiguous memory allocation is basically a method in which a single contiguous section/part of memory is allocated to a process or file needing it

First Fit

In the first fit approach is to allocate the first free partition or hole large enough which can accommodate the process. It finishes after finding the first suitable free partition.

Best Fit

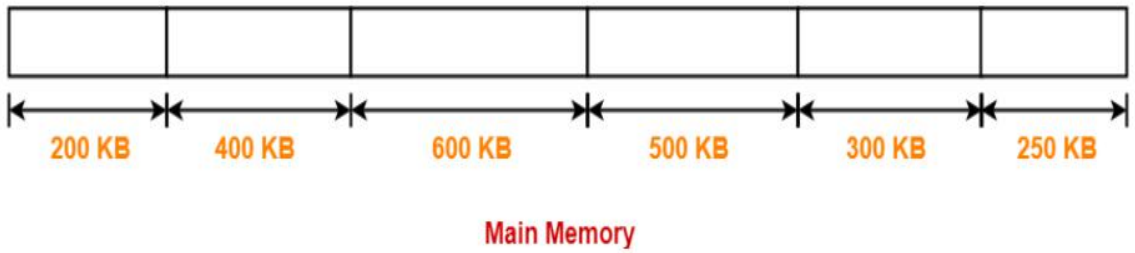
The best fit deals with allocating the smallest free partition which meets the requirement of the requesting process. This algorithm first searches the entire list of free partitions and considers the smallest hole that is adequate. It then tries to find a hole which is close to actual process size needed.

Worst fit

In worst fit approach is to locate largest available free portion so that the portion left will be big enough to be useful. It is the reverse of best fit.

Consider six memory partitions of size 200 KB, 400 KB, 600 KB, 500 KB, 300 KB and 250 KB. These partitions need to be allocated to four processes of sizes 357 KB, 210 KB, 468 KB and 491 KB in that order.

The main memory has been divided into fixed size partitions as-



Let us say the given processes are-

Process P1 = 357 KB
Process P2 = 210 KB
Process P3 = 468 KB
Process P4 = 491 KB

In First Fit Algorithm,

Algorithm starts scanning the partitions serially.
When a partition big enough to store the process is found, it allocates that partition to the process.

Step-01:

Step-02:

Step-03:

Step-04:

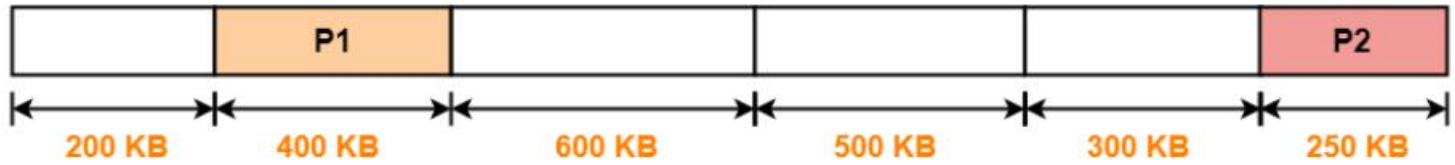
Process P4 can not be allocated the memory.
This is because no partition of size greater than or equal to the size of process P4 is available.

In Best Fit Algorithm,

Algorithm first scans all the partitions.
It then allocates the partition of smallest size that can store the process.

Step-01:

Step-02:



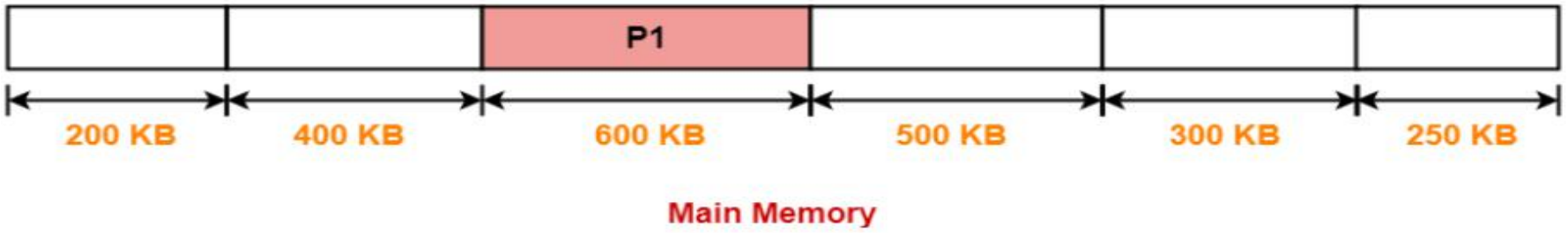
Step-03:

Step-04:

In Worst Fit Algorithm

Algorithm first scans all the partitions.
It then allocates the partition of largest size to the process.

Step-01:



Step-02:

Step-03:

Process P3 and Process P4 can not be allocated the memory.
This is because no partition of size greater than or equal to the size of process P3 and process P4 is available.

Operating Systems

Paging

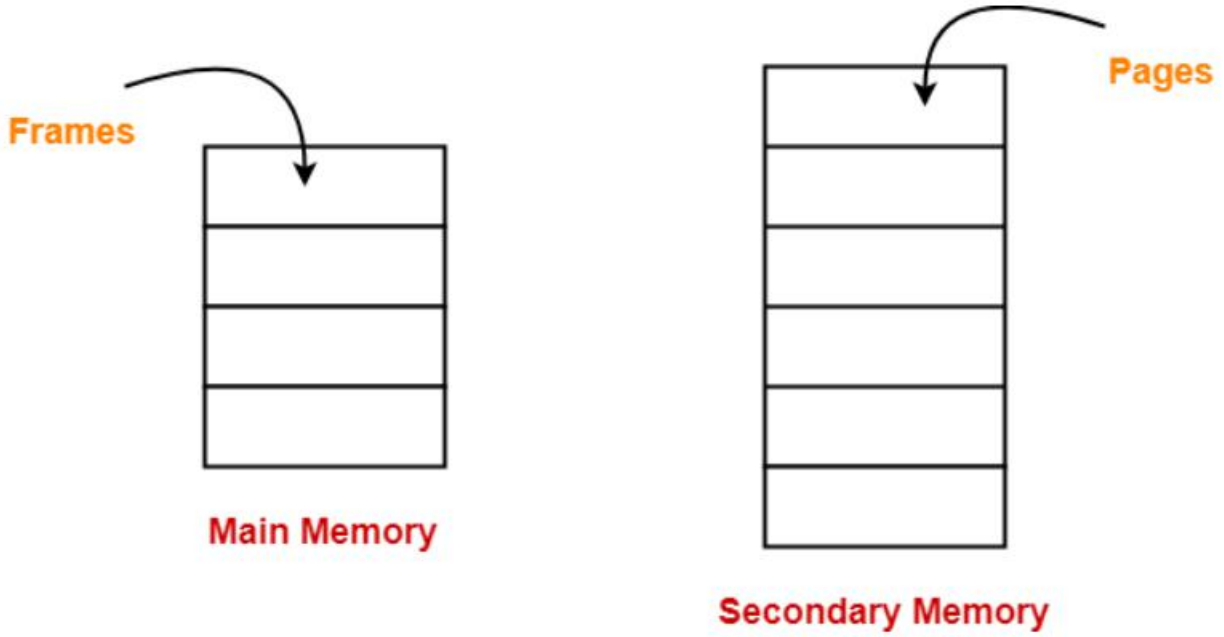
-: Done By

Saurabh Kishor
19CSXXX CSE2

Paging

Paging- Paging is a memory management technique in which the memory is divided into fixed size pages. Paging is used for faster access to data. so Paging is a fixed size partitioning scheme.

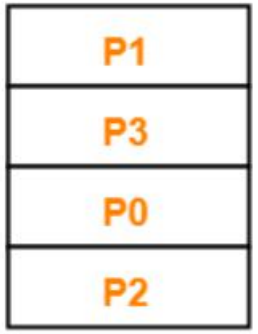
- In paging, secondary memory and main memory are divided into equal fixed size partitions.
- The partitions of secondary memory are called as pages.
- The partitions of main memory are called as frames.



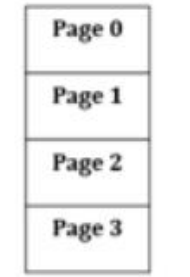
Example

- Each process is divided into parts where size of each part is same as page size.
- The size of the last part may be less than the page size.
- The pages of process are stored in the frames of main memory depending upon their availability.

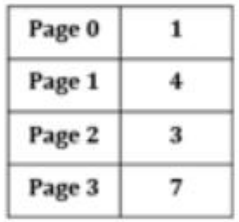
- Consider a process is divided into 4 pages P0, P1, P2 and P3.



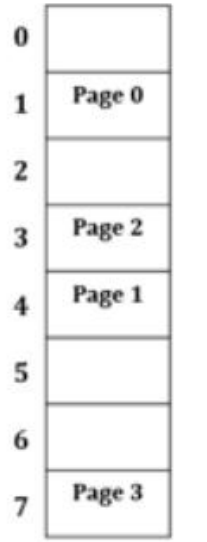
Main Memory



Logical Memory



Page Table

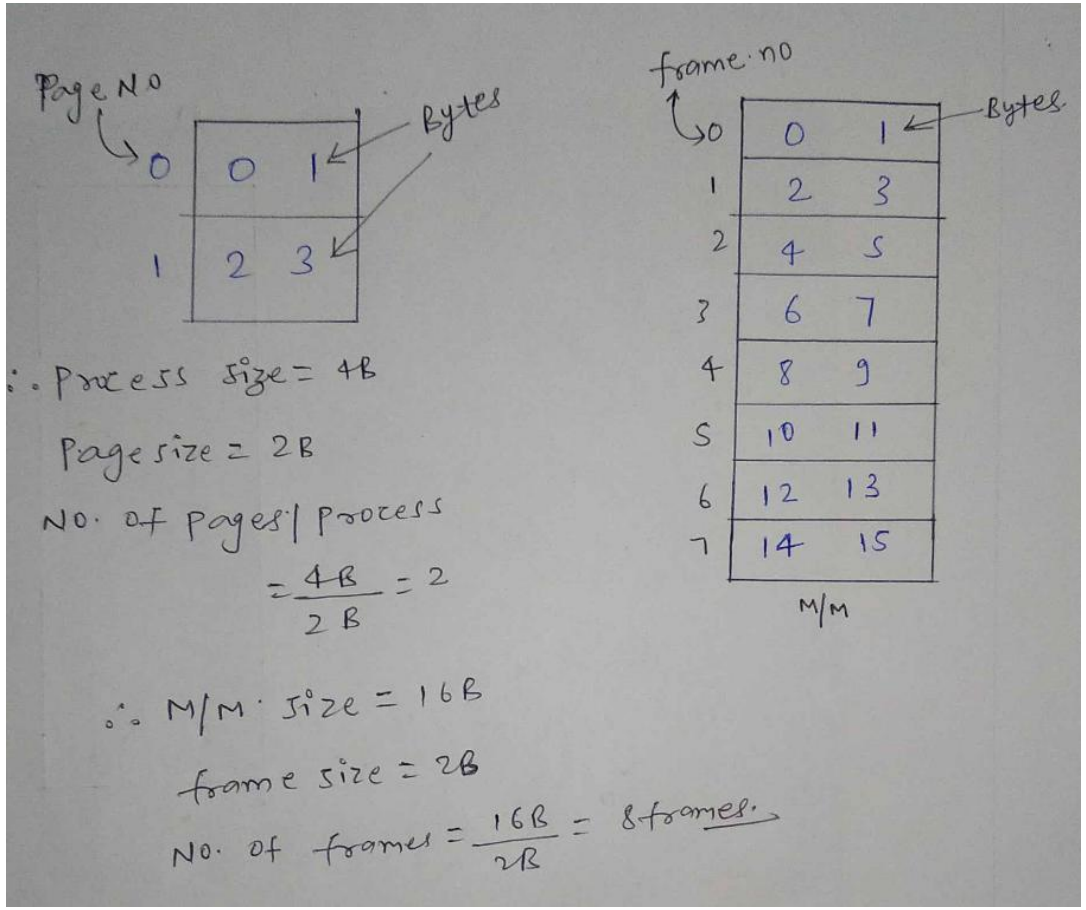


Physical Memory

Let's assume we have a process P1 whose process size is 4 bytes and page size is 2 bytes

So total size of the process, as we have assumed, is 4 bytes
size of a single page is 2 bytes
number of pages per process will be = 4/2 bytes
= 2 bytes
Now , the process P1 is split into two pages 0 and 1 in which the numbers 0, 1, 2 and 3

So there are 4 bytes in the process
We have 16 byte Main Memory
The number of frames can be calculated by dividing the memory size (16 Bytes) by the frame size (2 Bytes)



Why page size and frame size should be same?
if we divide a process into pages then one of the pages can be taken and fitted in the frame in compact
Means the page should fit in the frame as it is. So that's why we have taken the page size and frame size same always.

Advantages-

The advantages of paging are:

- It allows to store parts of a single process in a non-contiguous fashion.
- It solves the problem of external fragmentation.

Disadvantages-

The disadvantages of paging are-

- It suffers from internal fragmentation.
- There is an overhead of maintaining a page table for each process.
- The time taken to fetch the instruction increases since now two memory accesses are required.

-

Operating Systems

Segmentation

-: Done By

Saurabh Kishor
19CSXXX CSE2

Segmentation

Segmentation: Segmentation is non-contiguous memory allocation technique.
In segmentation, process is not divided blindly into fixed size pages.
Rather, the process is divided into modules for better visualization.

Characteristics
Segmentation is a variable size partitioning scheme.
In segmentation, secondary memory and main memory are divided into partitions of unequal size.
The size of partitions depend on the length of modules.
The partitions of secondary memory are called as segments.

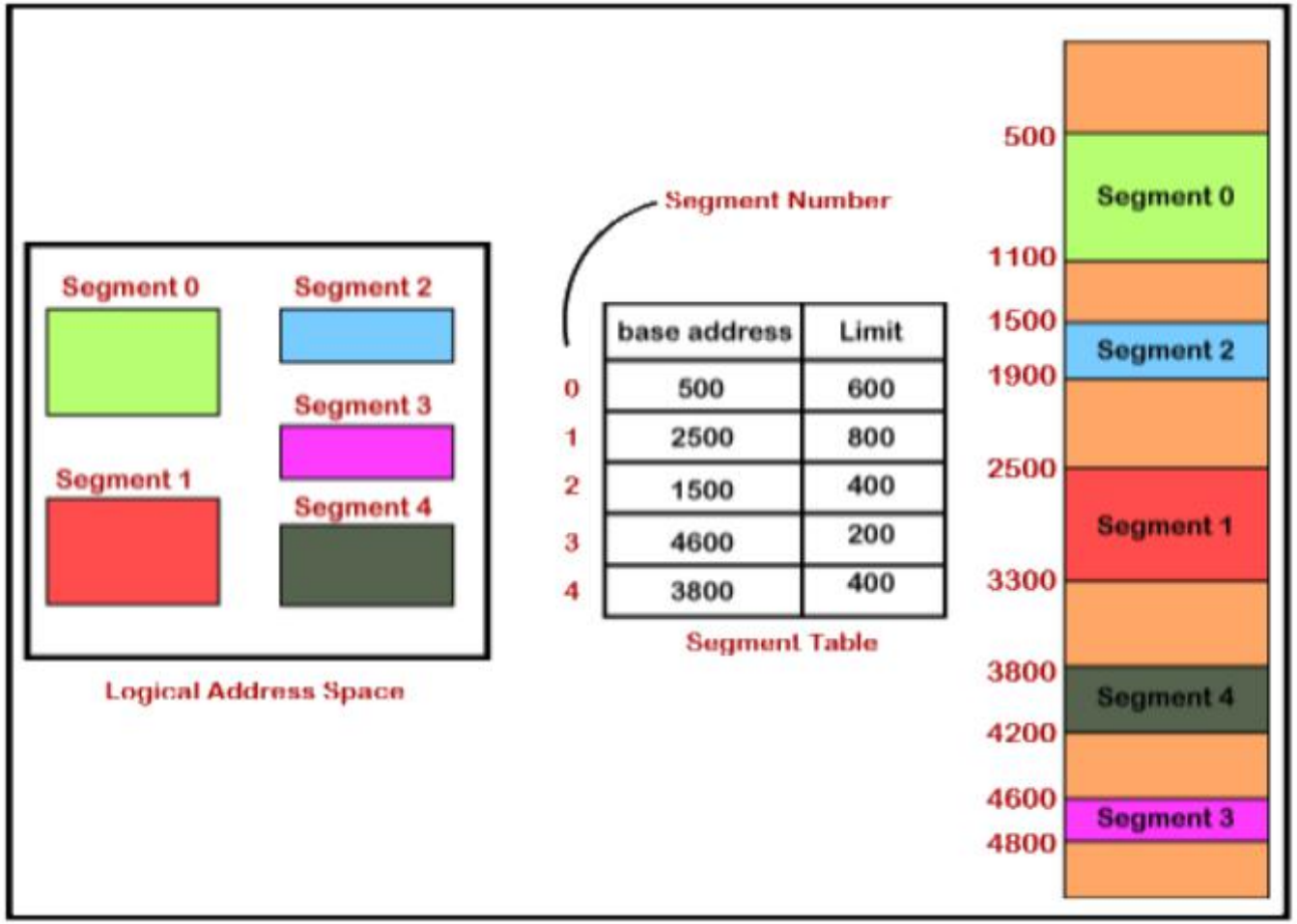
There are types of segmentation:

1. **Virtual memory segmentation**
   Each process is divided into a number of segments, not all of which are resident at any one point in time.
2. **Simple segmentation**
   Each process is divided into a number of segments, all of which are loaded into memory at run time, though not necessarily contiguously.

Segment table is a table that stores the information about each segment of the process.
It has two columns.

1. Limit: First column stores the size or length of the segment.

2. **Base address:** Second column stores the base address or starting address of the segment in the main memory.
Segment table is stored as a separate segment in the main memory.
Segment table base register (STBR) stores the base address of the segment table



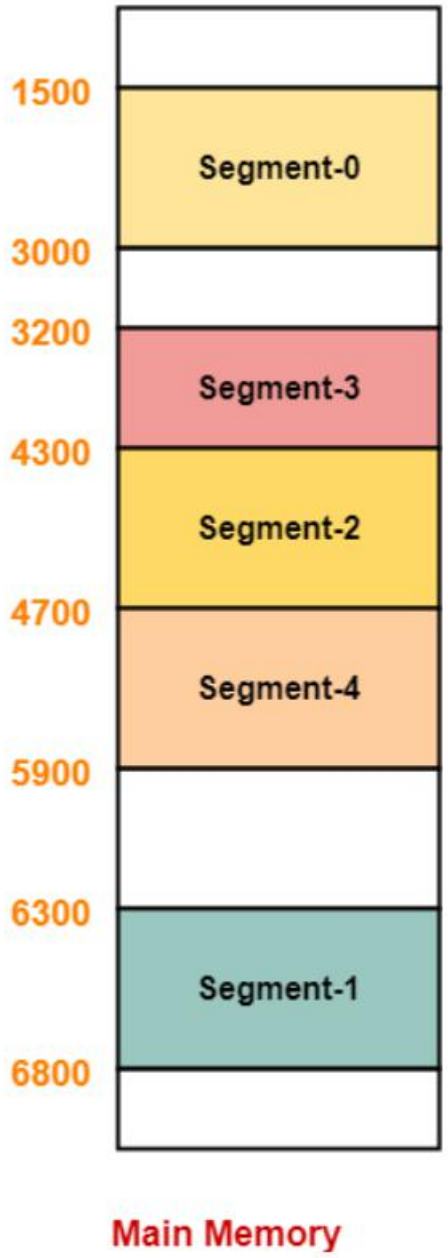
How to segments is stored physical memory ?

the segments are stored in the main memory as

first segment 0 so the segment 0 have the base address the which is starting address of the segment 0 that is 1400 and the length of the segment thousand so fourteen thousand plus thousand two thousand four hundred so the Sigma is zero stored.

next, size is three thousand two hundred that is a segment three take the segment three the starting address is 3200 and the limit the length of that is 1100 so just add 1100 to this so it becomes four thousand three hundred so the next address base address is 4,300 so here the segment two has to be stored. and the size of that segment is four hundred just for add four hundred to this 4,300 so then it becomes four thousand seven hundred and segment for that is the thousand so just add thousand to this it becomes five thousand so the starting address is four thousand seven hundred and ending of that segment is five thousand seven hundred.

this is way how each segment will be stored in the physical memory



Advantages of Segmentation

- It allows to divide the program into modules which provides better visualization.
- Segment table consumes less space as compared to Page Table in paging.
- It solves the problem of internal fragmentation.

Disadvantage of Segmentation

-
- There is an overhead of maintaining a segment table for each process.
- The time taken to fetch the instruction increases since now two memory accesses are required.
- Segments of unequal size are not suited for swapping.
- It suffers from external fragmentation as the free space gets broken down into smaller pieces with the processes being loaded and removed from the main memory.

Operating Systems

Page table

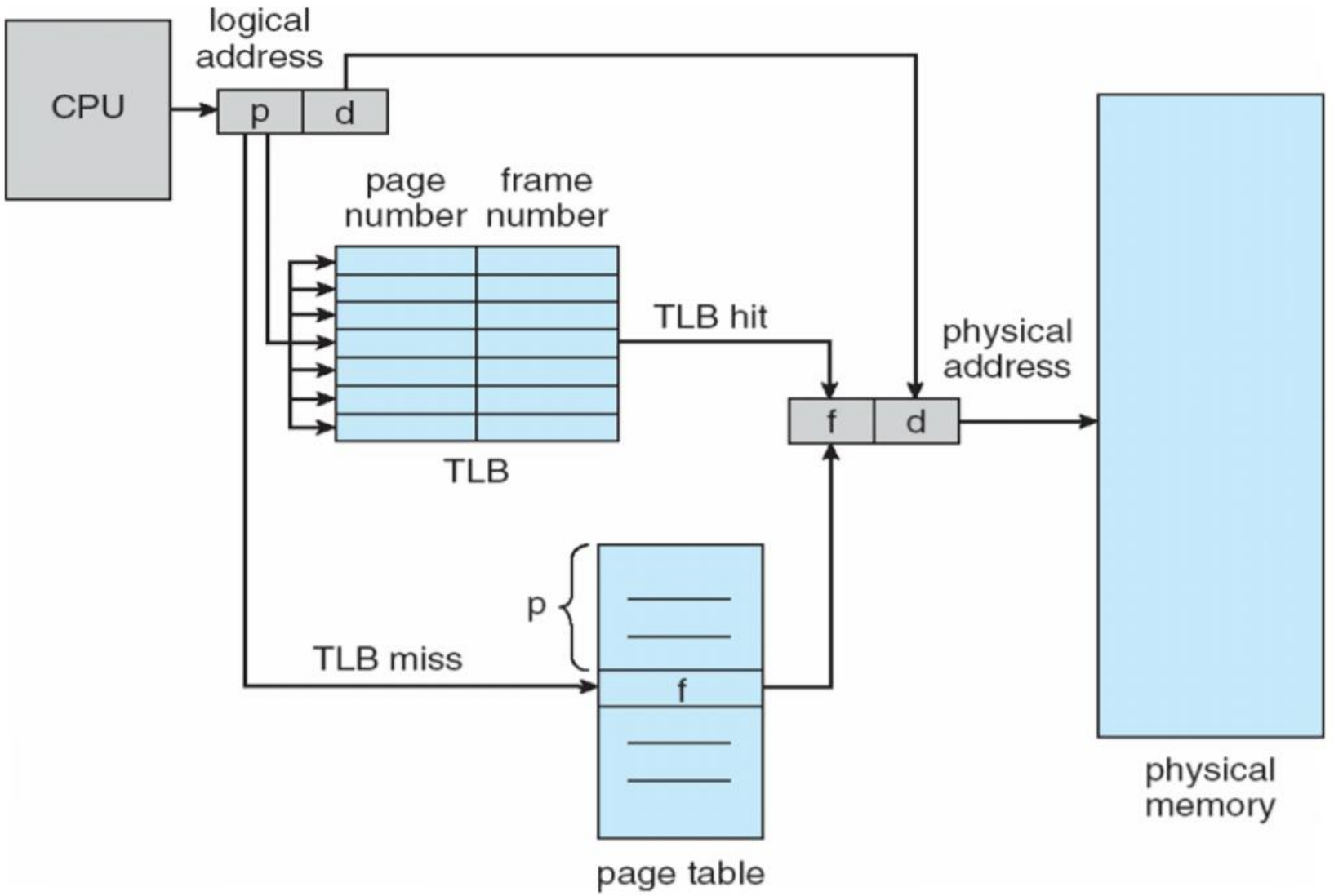
-: Done By

Saurabh Kishor
19CSXXX CSE2

Page table

Page Table: It is a data structure used by the virtual memory system to store the mapping between logical addresses and physical addresses.
The hardware implementation of page table can be done by using dedicated registers. But the usage of register for the page table is satisfactory only if page table is small. If page table contain large number of entries then we can use TLB(translation Look-aside buffer), a special, small, fast look up hardware cache.

The two memory access problem can be solved by the use of a special fastlookup hardware cache of associative memory called translation look-aside
buffers (TLBs)
- page number in TLB to quickly determine the frame number
• TLBs are typically small (64 to 1,024 entries)
• On a TLB miss, value is loaded into the TLB for faster access next time
- Replacement policies must be considered (which entries to overwrite when TLB is full)
- Some entries can be wired down for permanent fast access (entries for kernel code)


CPU
logical address
p
d
page number
frame number
TLB hit
physical address
f
d
TLB
p
TLB miss
f
page table
physical memory

Thank You