Chapter-8

Main Memory

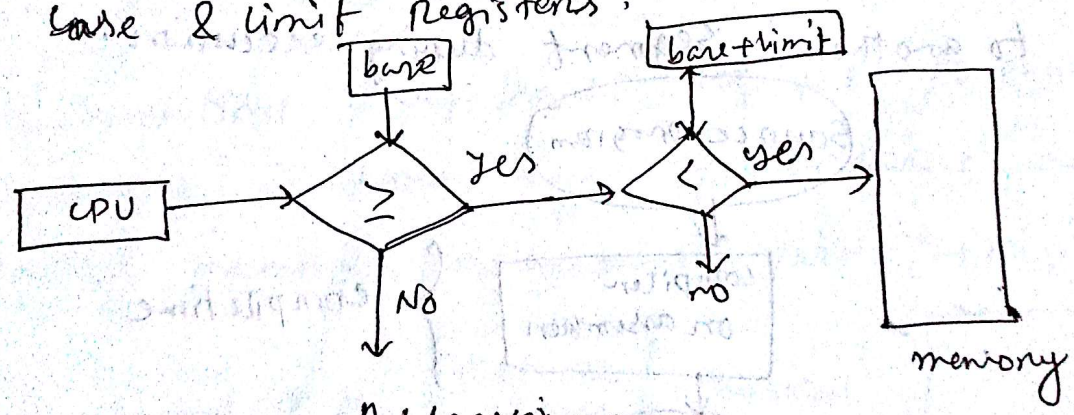
Base registers: It held smallest legal physical memory address.

Limit registers: It specify the size of range.

Let base register 400 & limit reg. 200. CPU can access all the address between 400 to 599.

How can ensure protection of address?

We can always check validity & protect address by use base & limit registers.



If an address requested by CPU is greater or equal base address and it is within the limit then the address is valid and it will access memory. Otherwise addressing error will occur.

Logical address: The address generated by CPU is called logical address or virtual address.

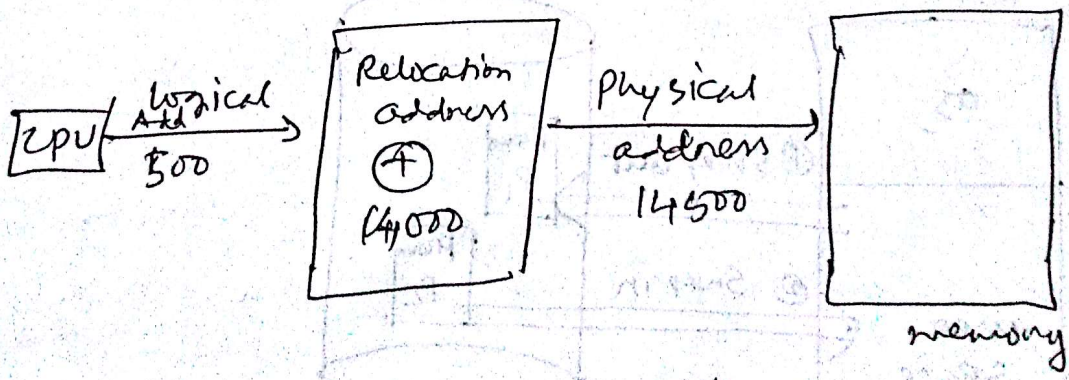
The set of all logical address is called logical address space.

Physical address: The address seen by the memory unit is called physical address. It is the actual address.

The set of all physical address ~~space~~ is called physical address space.

Write down about MMU :-

The runtime mapping of logical to physical memory address is done by a hardware device is called memory management unit (MMU).

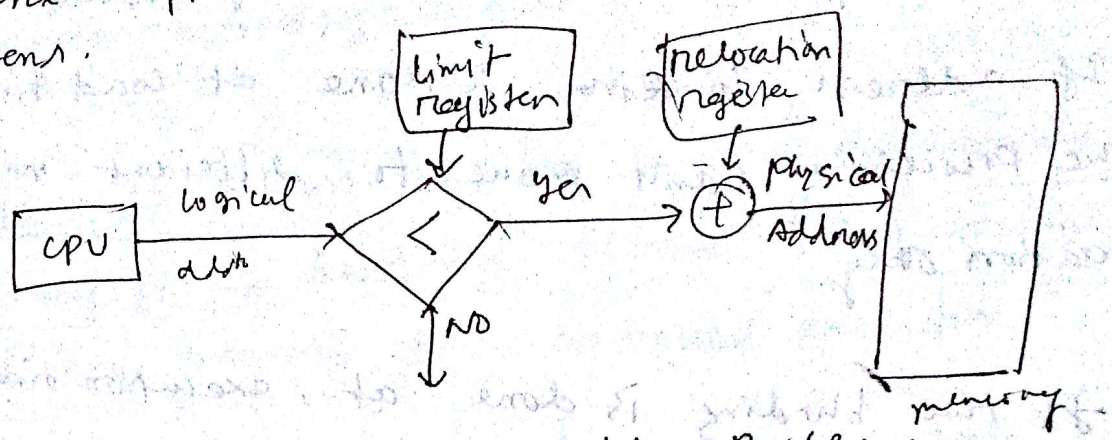


Adding logical address with Relocation register we get physical address of memory.

Static linking: system libraries & program code combined by the loader into the binary program image.

Hole → Block of available Memory

Hardware support for Relocation & Limit registers.



Relocation registers are used to Protect users Processes from each other. & from changing OS code & data.

Logical add. must be less than limit register

⇒ How to satisfy a target of size n from a list of free hole.

(i) First fit: Allocate the first hole that is big enough.

(ii) Best fit: Allocate smallest hole big enough.

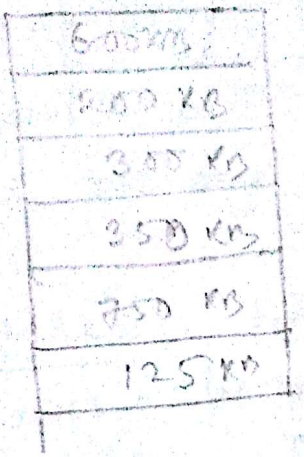
(iii) worst fit: Allocate the largest hole big enough.

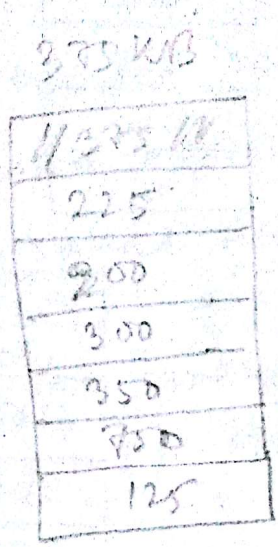
First & best fit is better than worst fit.

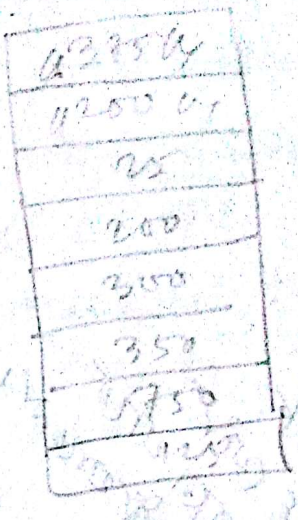
term of speed & storage utilization

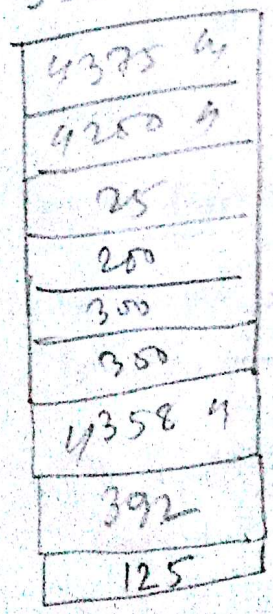
Here memory partitions are:
600 KB, 200 KB, 300 KB, 350 KB, 750 KB, 125 KB.

First fit:

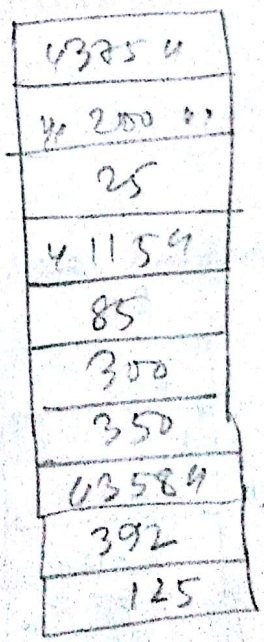








No place for 500KB



best fit

375 KB

650
200
300
350
750
125

375
225
200
300
350
750
125

375
225
200
300
350
750
125

375
225
200
300
350
358
392
125

500 KB No place

375
225
250
350
350
358
392
115
10

worst fit

375

650
200
300
350
375
375
125

210

650
200
300
350
375
375
125

358

358
242
200
300
350
375
375
125

115

358
242
200
300
350
375
375
260
125

Noplace 500

External Fragmentation:

External Fragmentation occurs when there enough total memory space exists to satisfy a request, but it is not contiguous.

Internal Fragmentation: Allocated memory may be slightly larger than requested memory. The difference between there memory is called internal Fragmentation.

50 Percent rule: First fit analysis reveals that given N blocks allocated, 0.5N blocks lost to fragmentation.

1/3 of memory may be unusable is called 50 percent rule.

Compaction:

External Fragmentation can be reduced by via compaction

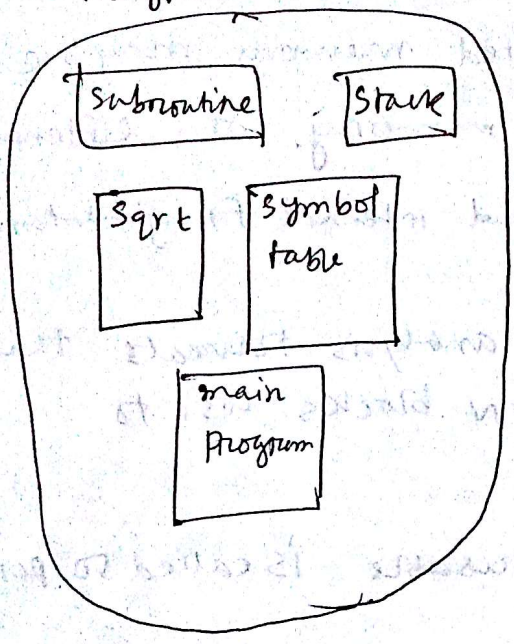
(i) Suffle memory contents to place all free memory together in one large block.

(ii) compaction is possible only if relocation is dynamic & is done at execution time.

Segmentation:

Segmentation is a memory management scheme that supports user view of memory.

A Program is a collection of segments.



Segmentation architecture:

① logical address space consists of two tuple <segment no, offset>.

② A segment table has two entry:-

(i) base: Indicate the starting of the physical address where the segments reside in memory.

(ii) Limit: Specify the length of the segment.

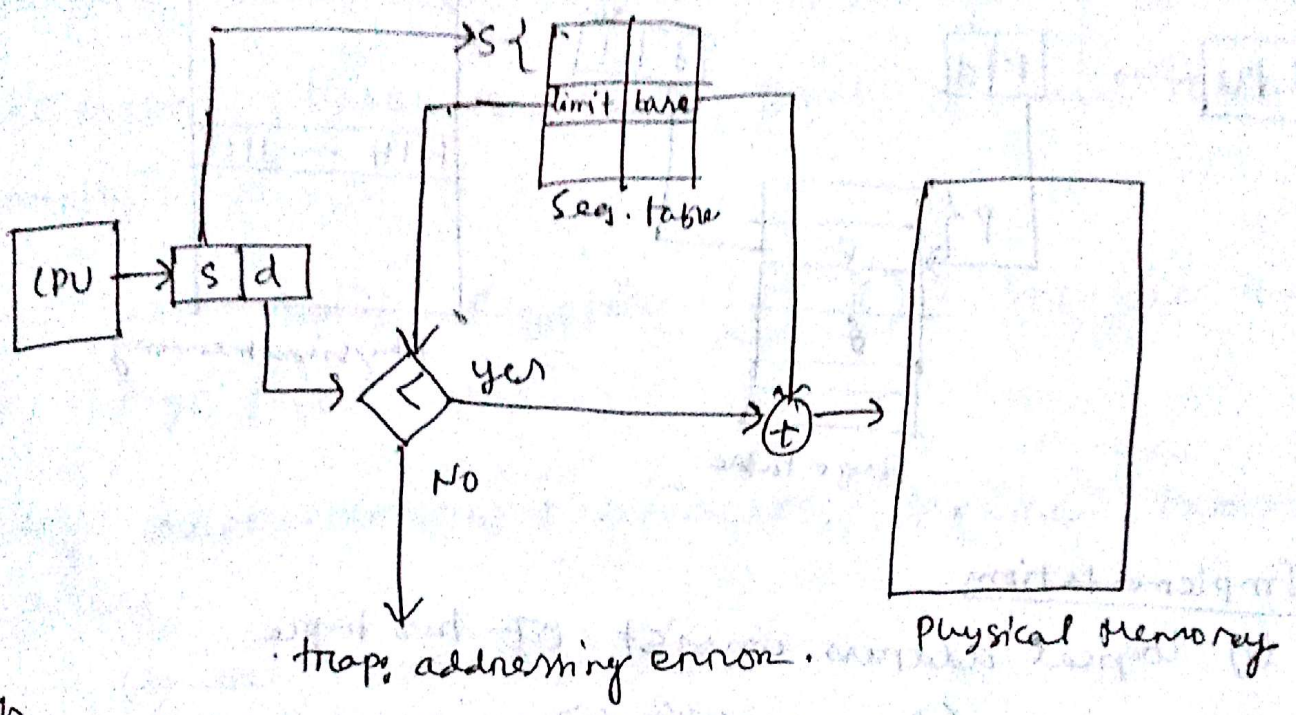
③ STBR: segment table base register points to the segment table's location in memory.

STLR: segment table length register indicates the number of segments used by a program.

Segment no. S is legal if $S < STLR$.

Validation bit = 0, illegal segment.

Validation bit = 1, legal segment.



Math 8.12

(a) 630 > 600 → trap in OS

(b) 13 < 14 ∴ so physical add = 2300 + 13 = 2313.

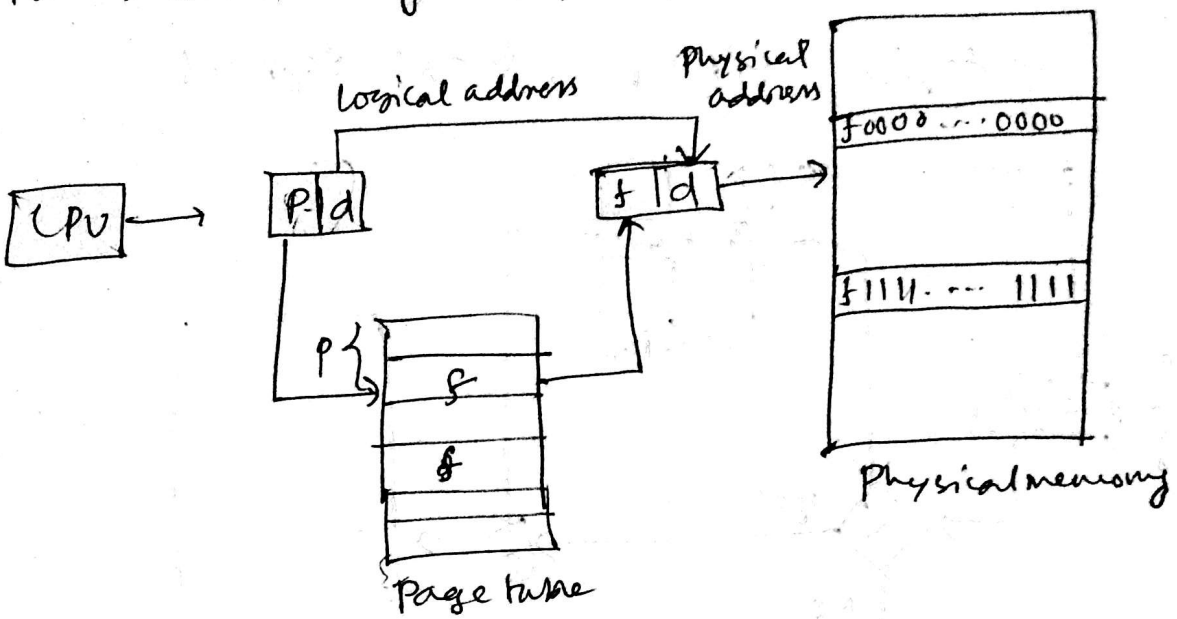
(c) 50 < 100 so " " = 90 + 50 = 140

(d) 581 > 580 so trap in OS.

(e) 92 < 96 so physical add = 1952 + 92 = 2044

Paging:

Paging is a memory management scheme by which computer stores & retrieve data from secondary storage to main memory. Physical address space to be non contiguous.



Implementation:

(i) Logical address consist of two tuple
<page no, Offset>

(ii) Page no. correspond a frame number.

(iii) By adding frame no. & Offset we get Physical address.

Frames: Divide the physical memory into fixed size block is called frames.

Page: Divide the logical memory into fixed size block is called page.

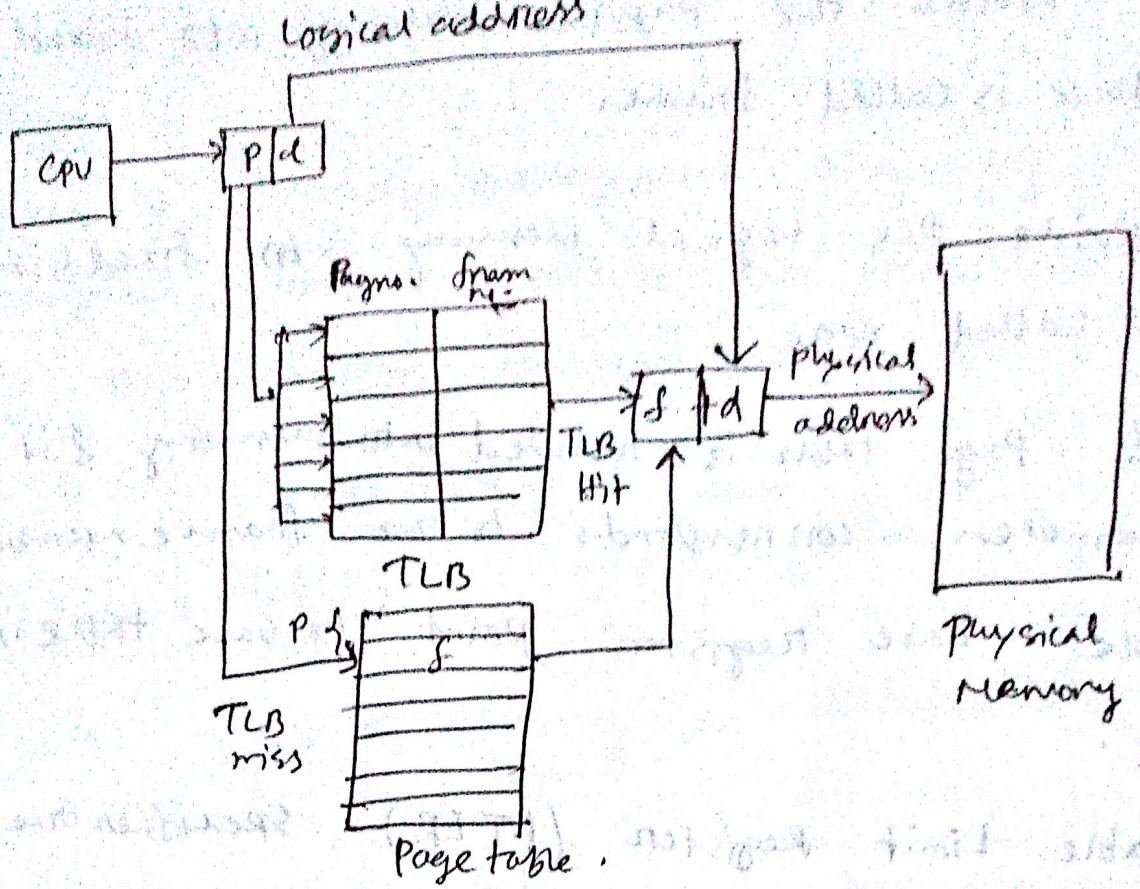
Page table: Page table is resided into memory & it has one parameter corresponds to the frame number.

Page table base register point to base table in memory.

Page table Limit Register (PTLR) - specifies the length of page table.

* Every data instruction access requires two memory access :-
  (i) one for page table.
  (ii) one for data/instruction.

TLBs: The two memory access problem can be solved by the use of a special fast look up hardware cache called Associative memory or translation look aside buffers (TLBs).



TLB contents:

A TLB consists of Page number & frame number.

If the Page no. in logical address found in TLB it is called TLB Hit, otherwise TLB Miss.

Actions:

(i) If TLB Hit occurs, get associative frame.

(ii) If TLB miss occurs get frame from memory & ~~put it on~~ TLB.

Effective Access Time (EAT):

Hit ratio = α

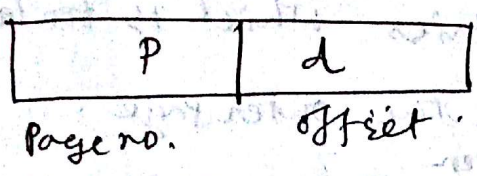
TLB access time = T.

memory access time = m.

Effective Access Time (EAT) = $(m+T)\alpha + (2m+T)(1-\alpha)$.

Why Page size is always power of 2?

The page address is divided into 2 part.



If there are m bit in logical address, n bit will offset and (m-n) bit page number.

Page size always power of 2 because it is easy to break into page number & page offset

Protection:

Memory protection implemented by associating protection bit with each frame. This protection bit is called valid, invalid bit.

(i) valid indicates that the associated page is in process.

(ii) invalid indicates that the page is not in process.

2010
Feb
8.9

(a) Paged memory reference take = 200+200 = 400 ns.

(b) E.A.T when TLB used = $(m+T)\alpha + (2\times m+T)(1-\alpha)$

$= (200+20)\times 0.75 + (2\times 200+20)\times 0.25$

$= 270$ ns.

2014
Sep

(i) Physical address of c = 7×4+2 = 30

(ii) " " g = ~~6×4+2~~ 5×4+2 = ~~26~~ 22

(iii) " " i = 2×4+0 = ~~10~~ 8

(iv) " " a = 7×4+0 = 28

(v) " " l = 2×4+3 = 11

(vi) " " n = 3×4+1 = 13

2^4 2^8

2012
6(a)

No. of Page = 8 = 2^3

No. of word = 1024 = 2^{10}

No of frame = 32 = 2^5

logical address bit = 3+10 = 13 bit

physical address bit = 5+10 = 15 bit.

Ch-8+1=9

Virtual memory

V.M. is a technique that allows the execution of processes that are not completely in memory.
It is separation of user logical memory from physical memory.

Advantage :-

1. only a part of program need to be in memory for execution
2. logical address space much larger than physical address space
3. Allow address space to be shared by others.
4. More program running concurrently.
5. Allow more efficient process creation.

Drawback :-

1. It is not easy to ~~implem~~ implement
2. Decrease performance if it is not use carefully.

Performance via Page fault:

Let, Page fault rate = p $\quad 0 \le p \le 1$

Now, p=0, No page fault.

p=1, every reference is a fault.

Effective Access Time (EAT)

$\therefore$ EAT = (1−p) x memory access + p(P.F overhead + swap in + swap out)

Performance of computer system is inversely proportional to page fault rate.

Ex:

Memory Acctime = 200 ns.

Page fault service time = 8 milisec

= 8×10^6 n

EAT = (1−p) x 200 + p(8×10^6)

= 200 − 200p + 8×10^6 p.

= 200 + 7999800 p.

Page fault rate $p = \frac{1}{1000}$

EAT = ~~2000~~ + 8×10^6

= 8.2 Microsec.

slow down factor = $\frac{8199.8}{200} \approx 40$.

brought into memory several time.

Page replacement:

Page replacement is the process of replacing page from memory which is not really used by currently demanded page.

steps in Page replacement:

1. Find the location of desired page on disk.
2. Find a free frame:-
   1. If there is a free frame, use it.
   2. If there is no free frame, use page replacement algo to select a victim frame.
   3. Write victim frame to disk if dirty.
3. Bring the desired page into the newly free frame, update the page & frame table.
4. continue the process by restarting the instruction that caused the trap.

Frame allocation algorithm :

determines.

(i) How many frames to give each process.

(ii) which frame to replace.

Page replacement algo :-

(i) compute the number of Pagefault on a reference string

(ii) want lowest page-fault rate on both first access & re-access.

There are three types of page replacement :-

1. FIFO (যেটা আগে error করেছে সেটা বের হয়ে যাবে)

2. Optimal (যেটা পরে (বেশি দেরি করে) বের হবে)

3. LRU (~~যেটা error~~ Present page থেকে যেটা বেশি সময় বের হয়নি error বের হবে)

Belady's Anomaly: In FIFO Page replacement Algorithm increase the no. of frame size will ~~decrease~~ increase no. of page fault.

This phenomena is called Belady's Anomaly

2015
6(a)

5	6	7	0	7	1	7	2	0	1	7	2	0	6	3	4	3	0	1	4	2
5	5	5	0		0		0			7		7	7	3	3		3	1		1
6	6	6	6		1		1			1		0	0	0	4		4	4		2
7		7	7		7		2			2		2	6	6	6		0	0		0
5	5	5	0		0		0			0		0	0	0			1			1
6	6	6	6		1		1			1		1	1	4			4			4
7		7	7		7		2			7		6	3	3			5			2
6	5	5	0		0		2	2	2	7		6	6	6		0	0	0		2
	6	6	6		1		1	0	0	0		0	4	4		4	1	1		1
		7	7		7		7	7	1	1		1	3	3		3	3	4		4

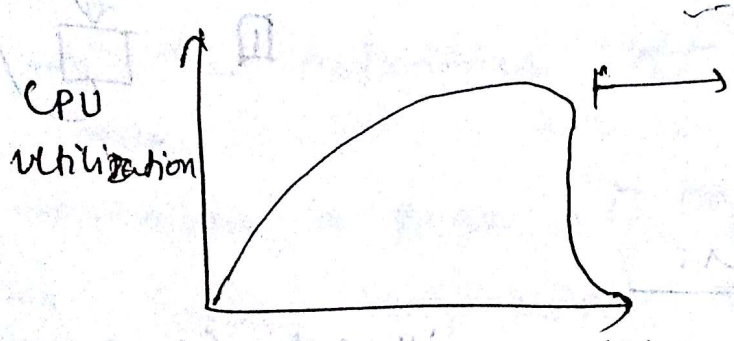
a_i = allocation size of p_i = $\frac{s}{m} \times m$.

Thrashing: A high ~~Priority~~ paging activity is called thrashing. A process is thrashing if it is spending more time paging than executing.

Thrashing results in several performance problem
To limit the effects of thrashing we need to.

(i) increase cpu utilization.

(ii) decrease the degree of multiprogramming.



(iii) If degree of multiprogramming is increased it will increase CPU utilization ~~too~~ until maximum is reached.

(iv) If degree of multiprogramming increased

Chapter-5

Critical section:

Consider a system consisting of n processes (P_0, P_1, P_2 ..... P_n) each process has a segment of code called critical section in which process may change their common variable, updating table, word files etc.

```
do {
    entry section
        critical section
    exit section
        remainder section
} while (TRUE);
```

```
do {
    while (turn == j)
        critical section
    Turn = j;
        remainder section
} while (true);
```

When a process is executing in critical section no other process can execute in critical section.

Critical section problem is a design protocol to solve this.

Requirment to critical section problem:

① **Mutual exclusion:** If process P_i is executing in its critical sections, then no other processes can be executing in their critical sections.

② **Progress:** If no process is executing in its critical section there exists some process wish to enter their critical section, then the selection of the processes that will enter

of time that other processes are allowed to enter their critical section before the request is granted.

Peterson's Solution:

A Peterson solution requires two data items to be shared between two process.

```
int turn;
boolean flag [2];
```

① The variable turn is used to determine which program will enter critical section.

```
do {
    flag [i] = true;
    turn = j;
} while (flag [j] && turn == j);
        critical section
    flag [i] = false;
        Remainder section
} while (true.
```