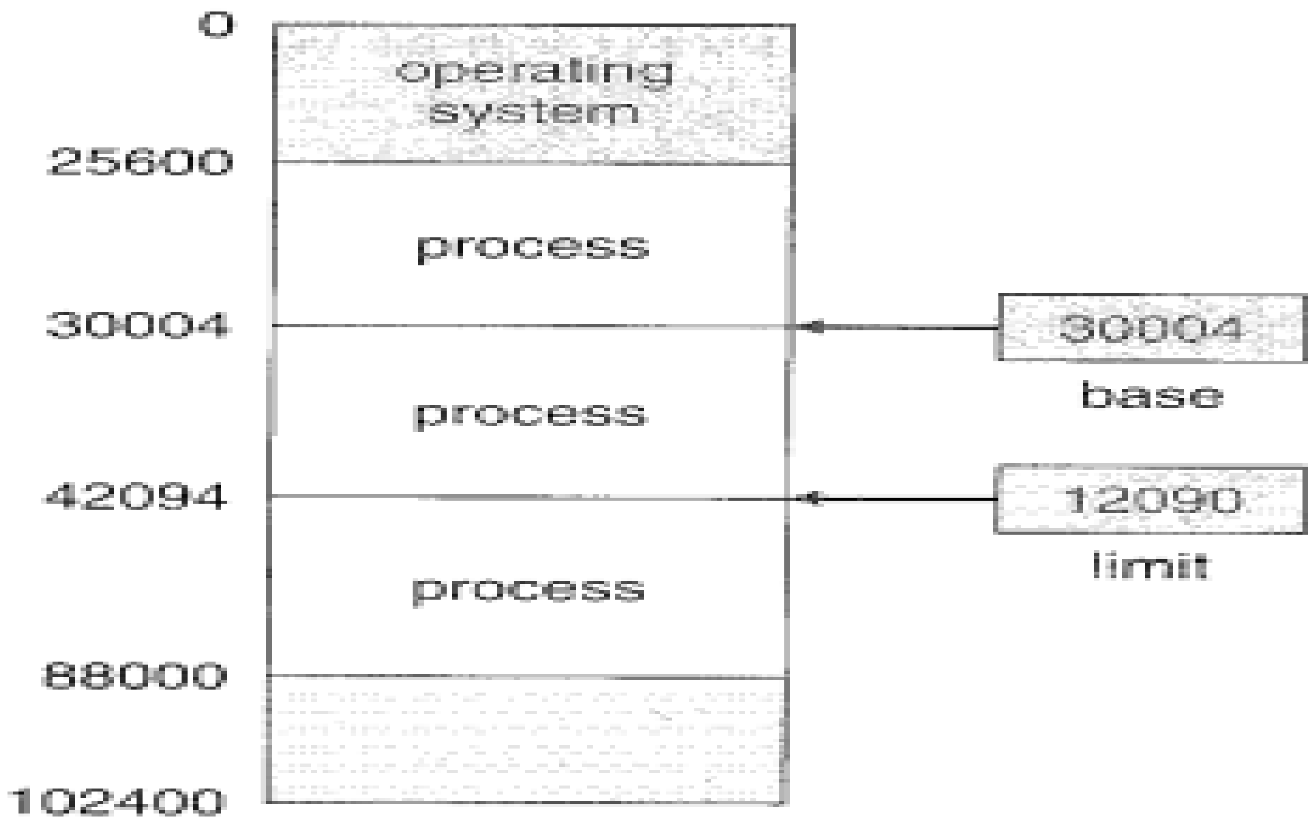
Memory Management

Dr. Padmaja Joshi

- *Every process is given a separate memory address which can be represented by **Base** and **Limit;***
- ***Limit*** *specifies the size whereas **Base** gives the starting physical address of the memory;*
- *The processes can access only those memory locations that fit in this range;*

Memory allocation for processes



Allocation

Fixed Size

Variable Size

Memory allocation

Every partition is allocated to a process. Only one process at any given instance of time.

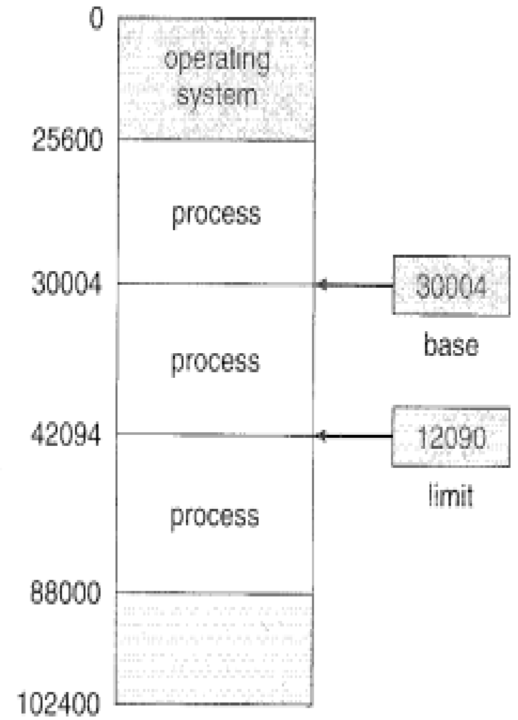
After allocation whichever partitions are available are called as holes.

The hole size will be variable or fixed depending upon the allocation method used.

Operating system maintains the list of holes along with their sizes.

Continuous Memory allocation

- *Usually memory is divided into two partitions: one for resident OS programs and other for user programs.*
- *There is a choice of keeping in the low memory or high memory. Since the interrupt vectors are there in the lower memory the OS is stored in the low memory.*
- *For allocation, memory is divided into varying size blocks to cater the need of different types and hence size of processes.*



Memory Allocation Algorithms

First Fit

- *A process is allocated to the first partition in the list that is big enough to accommodate it.*

Best Fit

- *Most appropriate sized partition is selected from the list to allocate it to the process*

Worst Fit

- *Largest partition is selected and allocated to the process.*

Example – First Fit, Best Fit Worst Fit

Memory partitions available – 300K, 600K, 350K, 200K, 750k, 125K

Processes – 115K,500K, 358K, 200K, 375K

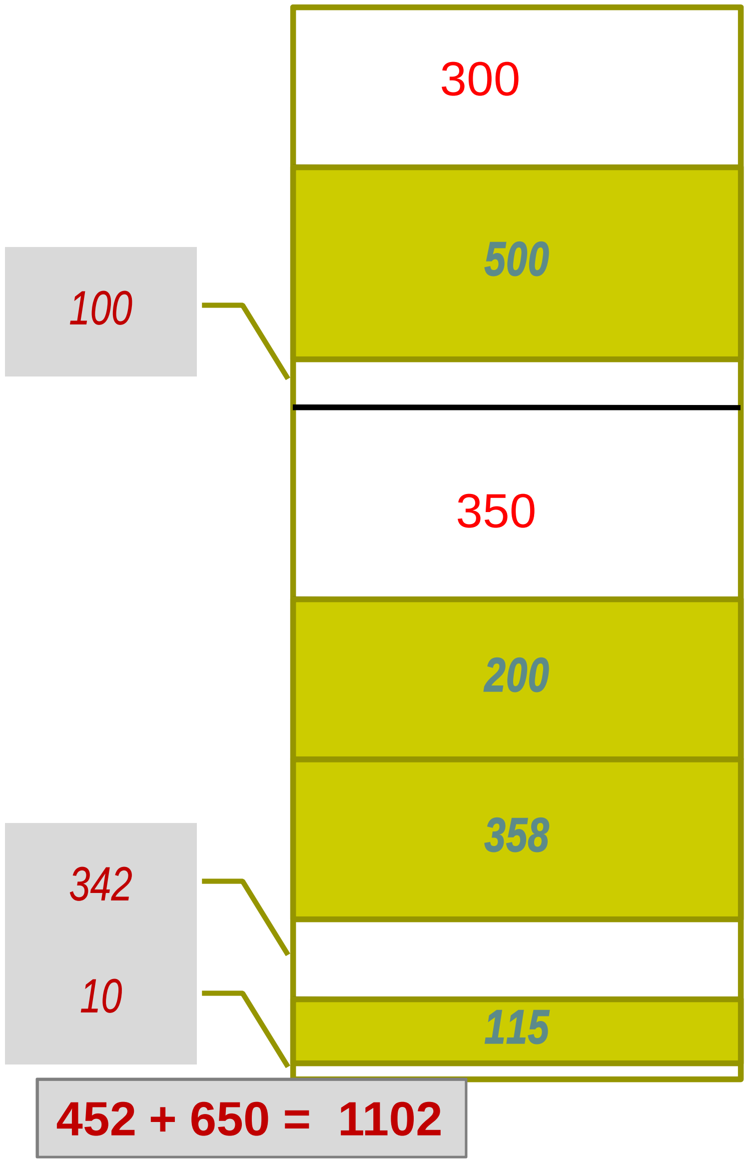
115K , 500K, 358K, 200K, **375K**

FIRST FIT

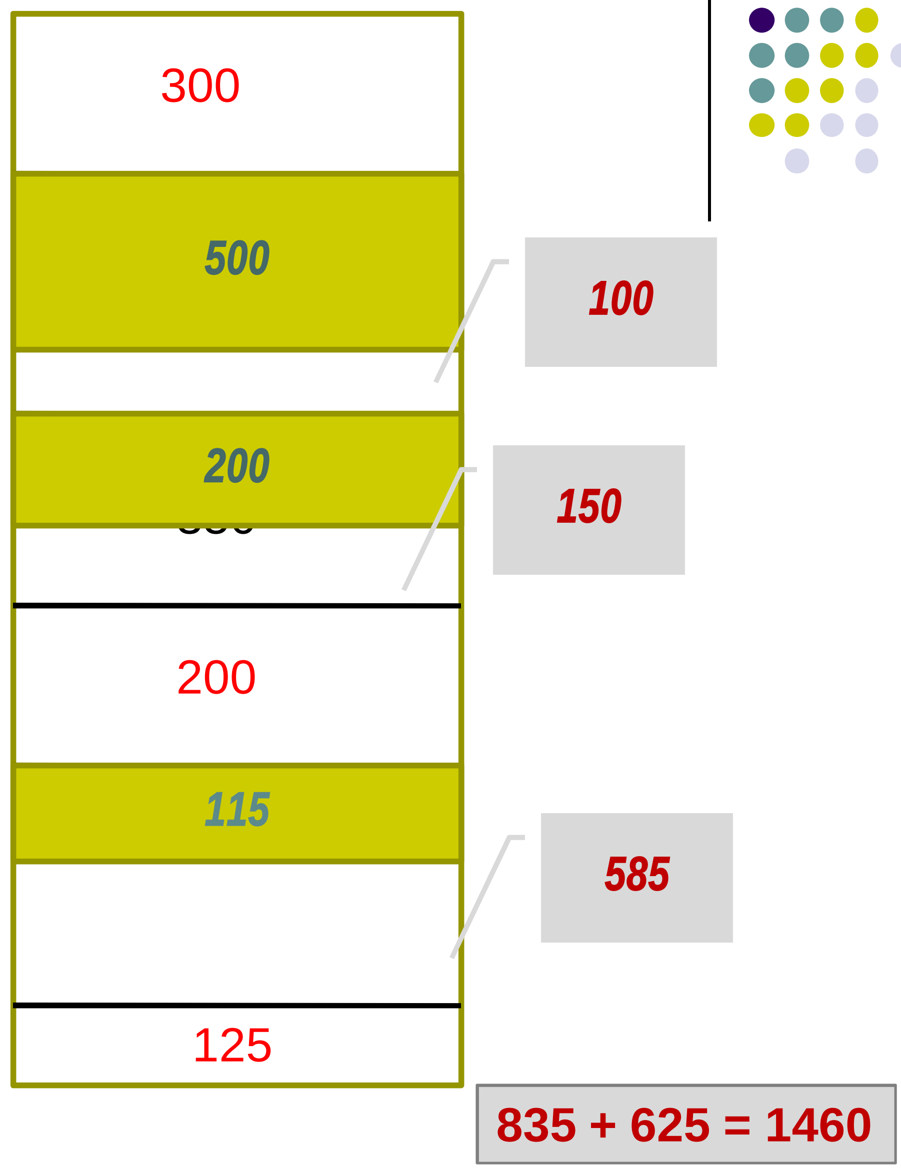
677 + 325 = 827

115K , 500K, 358K, 200K, 375K

BEST FIT



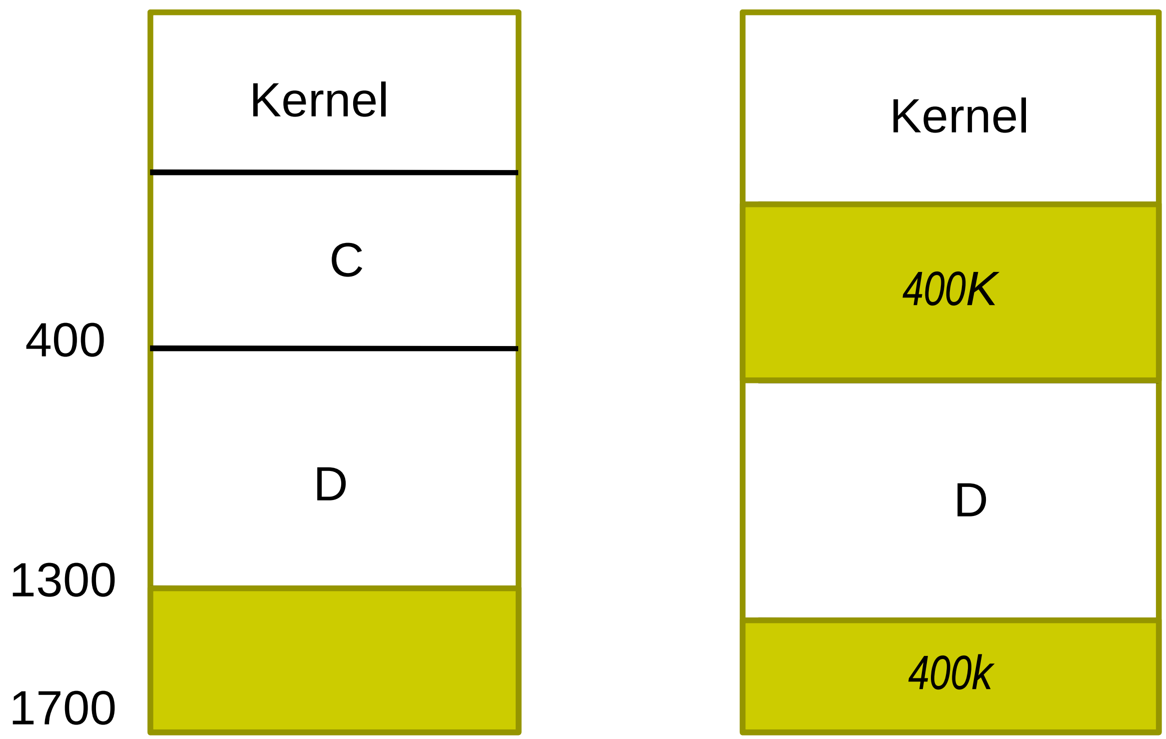
WORST FIT



Example

Program	Size
C	400 K
D	900K
E	550K
F	700 K

Assume that the system has 1.7M free for programs.



Fragmentation

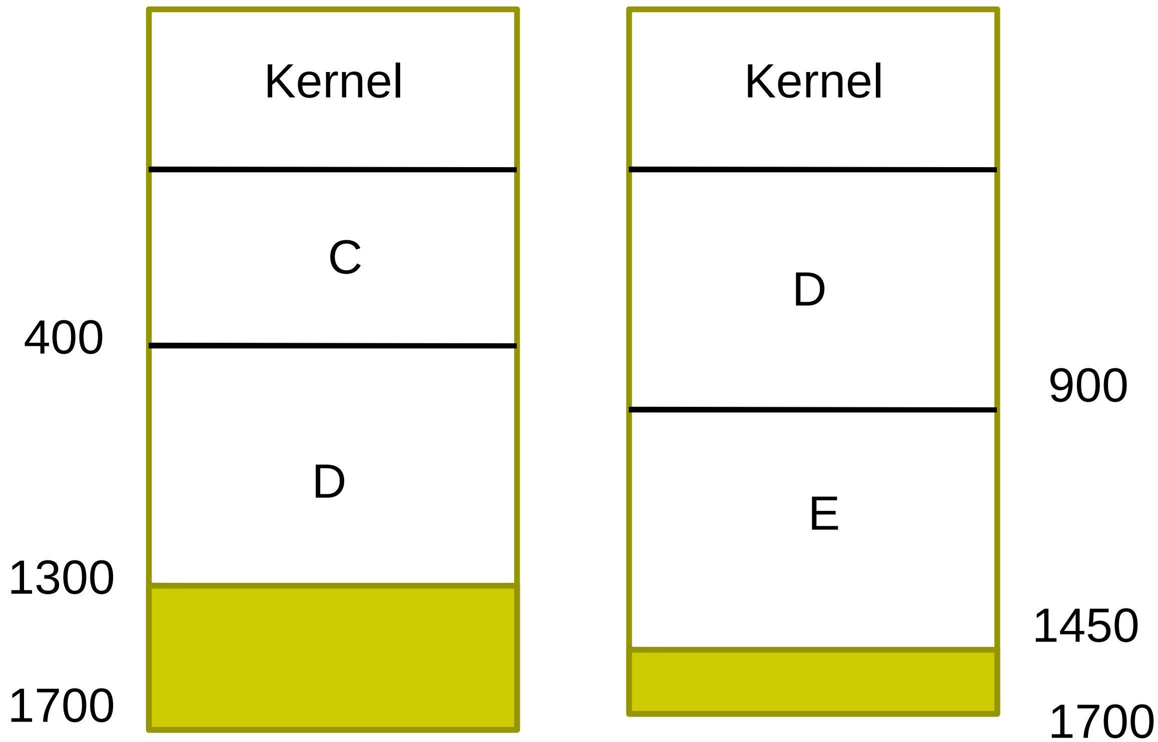
Inefficient memory utilization results in Fragmentation

- *External*
  - *Enough memory but not continuous*
  - *Compaction*
- *Internal*
  - *Allocated large memory and hence unutilized*

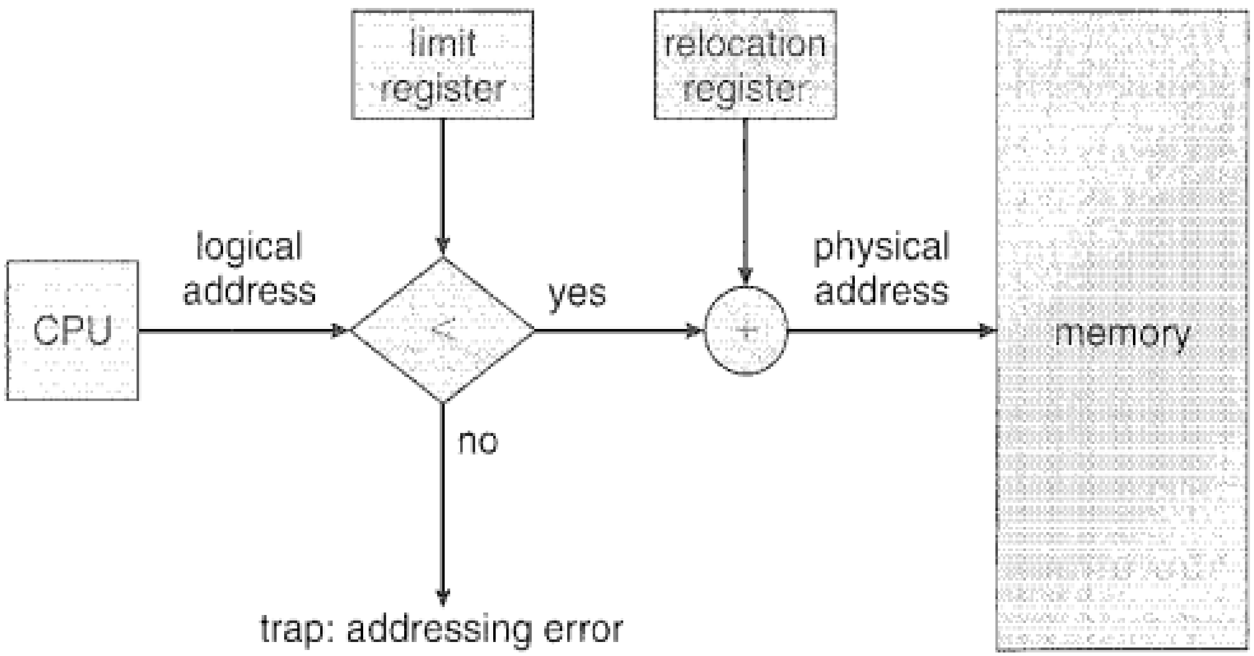
Point	***Internal Fragmentation***	***External Fragmentation***
Definition	*When the allocated memory is more than the required memory, it gives rise to internal fragmentation.*	*In the memory, there are small but un-continuous blocks available if combined they can suffice the memory need of the next program.*
Memory Size	*Occurs when memory is divided into fixed sized blocks.*	*Occurs when the blocks allocation is of varying size.*
Solution	*Best Fit allocation*	*Compaction*

Compaction

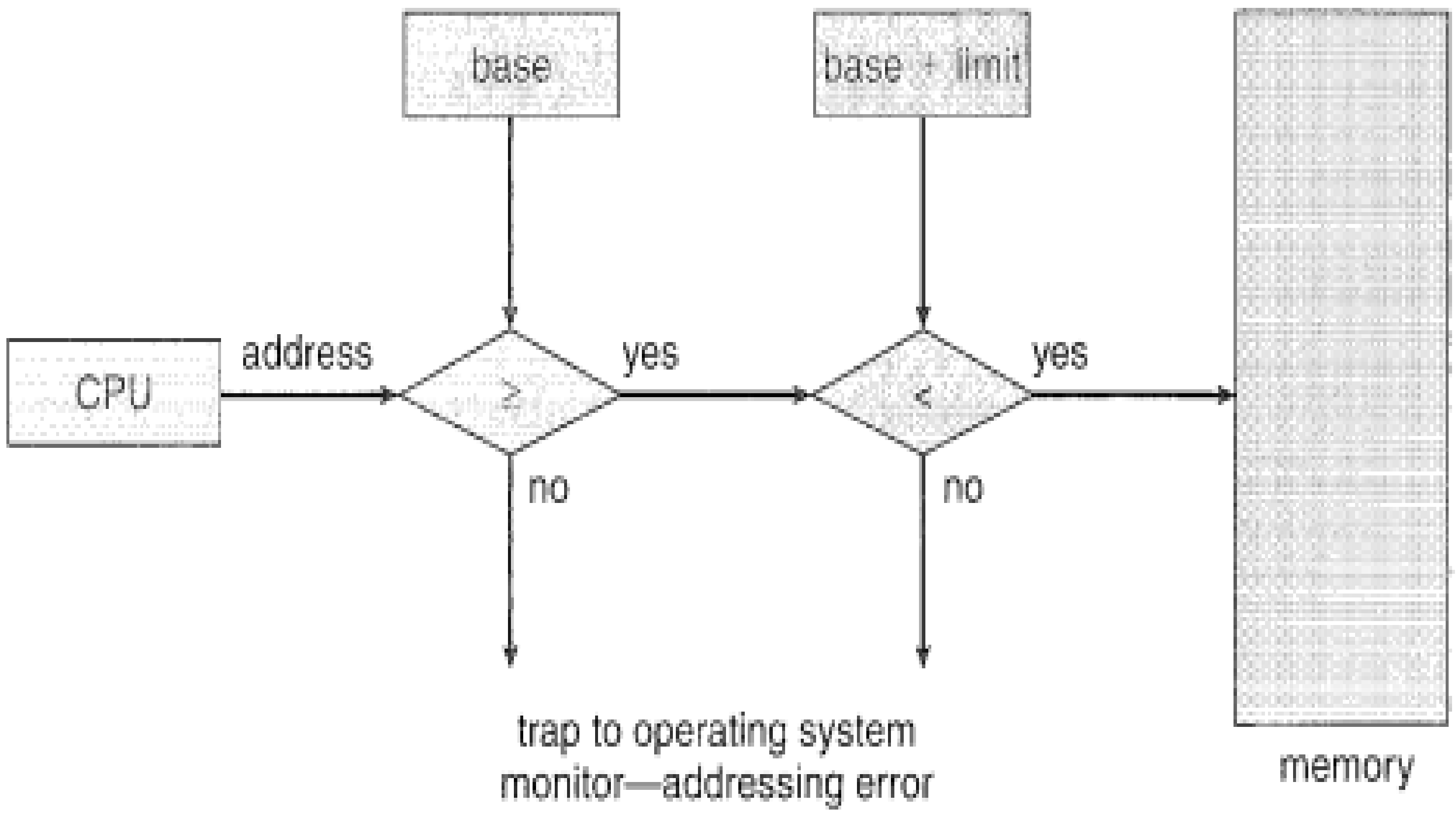
- *Moving the program for one memory location to another in order to get the continuous memory is called as "Compaction".*



Hardware support for Dynamic Allocation



Hardware Support for Memory Access Protection



Paging

Memory is divided into small fixed sized blocks called as frames and the process is divided into pages.

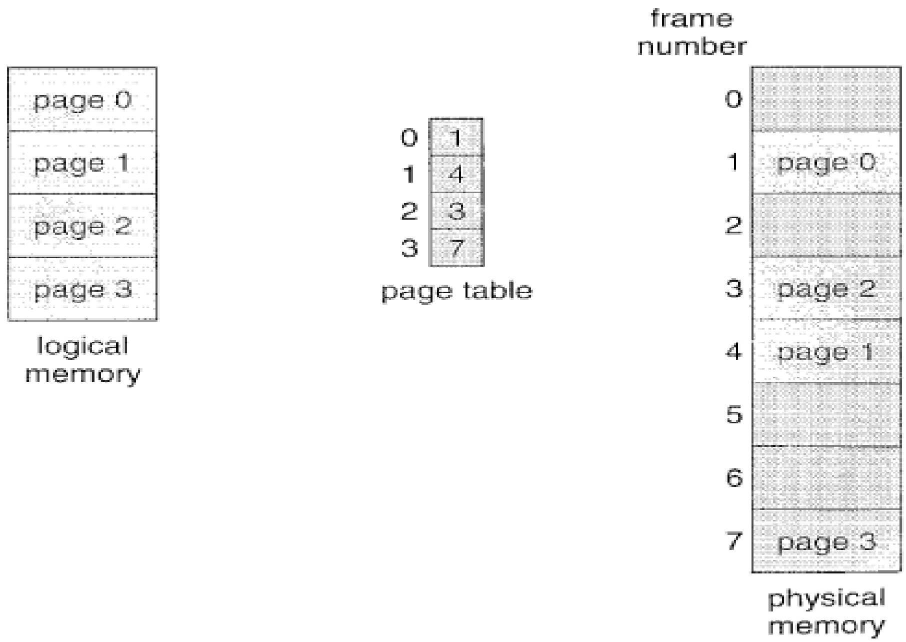
The size of the page is a power of 2 varying between 512 bytes to 1 GB

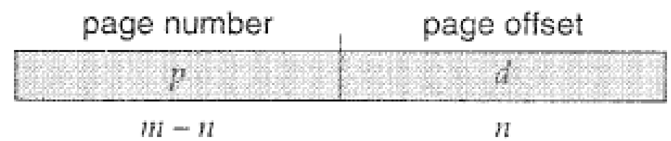
- *Typically is it 4KB to 8 KB*

To get the page size in the system

- *getconf PAGESIZE*
- *or use a system call getpagesize();*

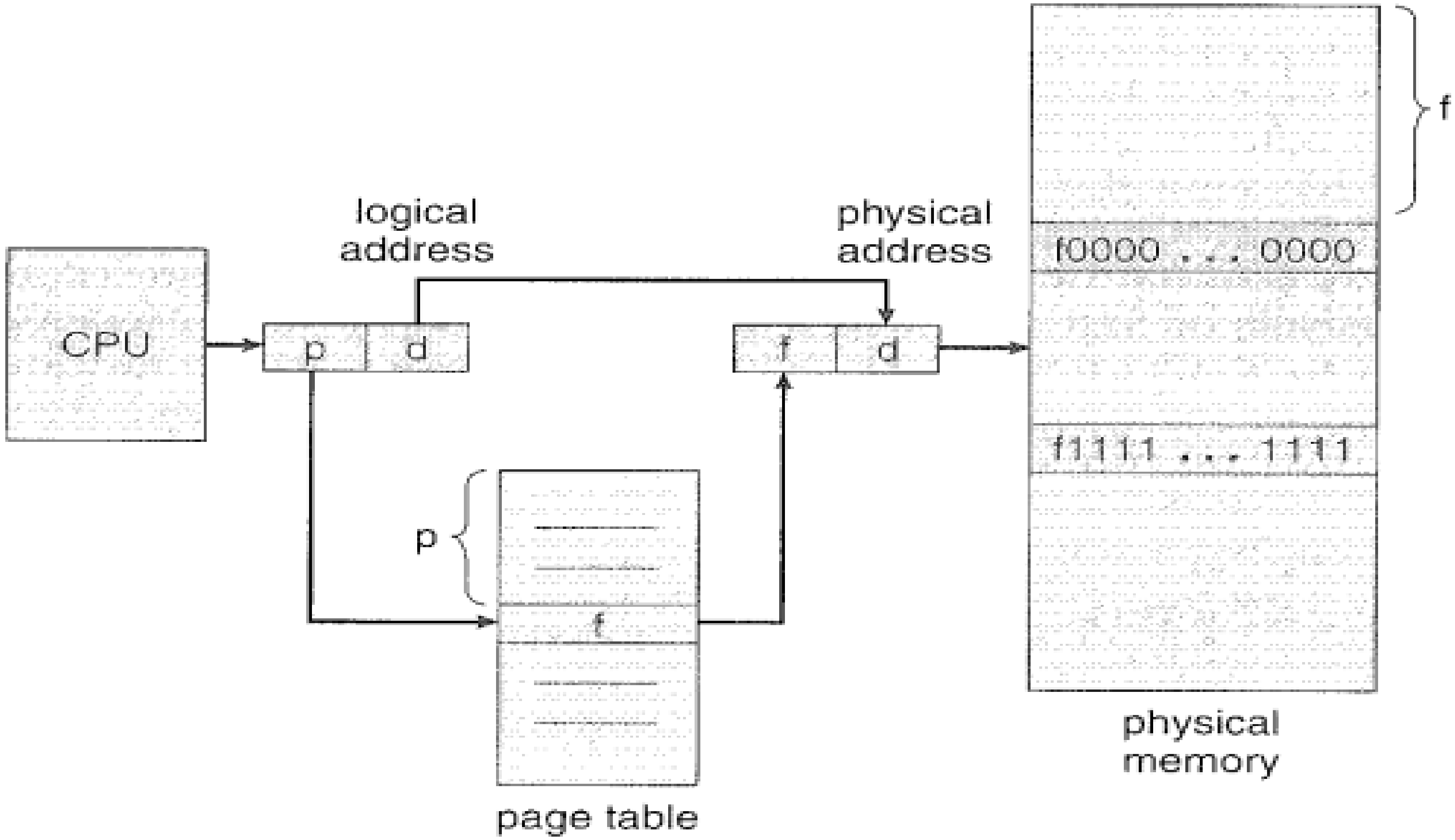
Paging





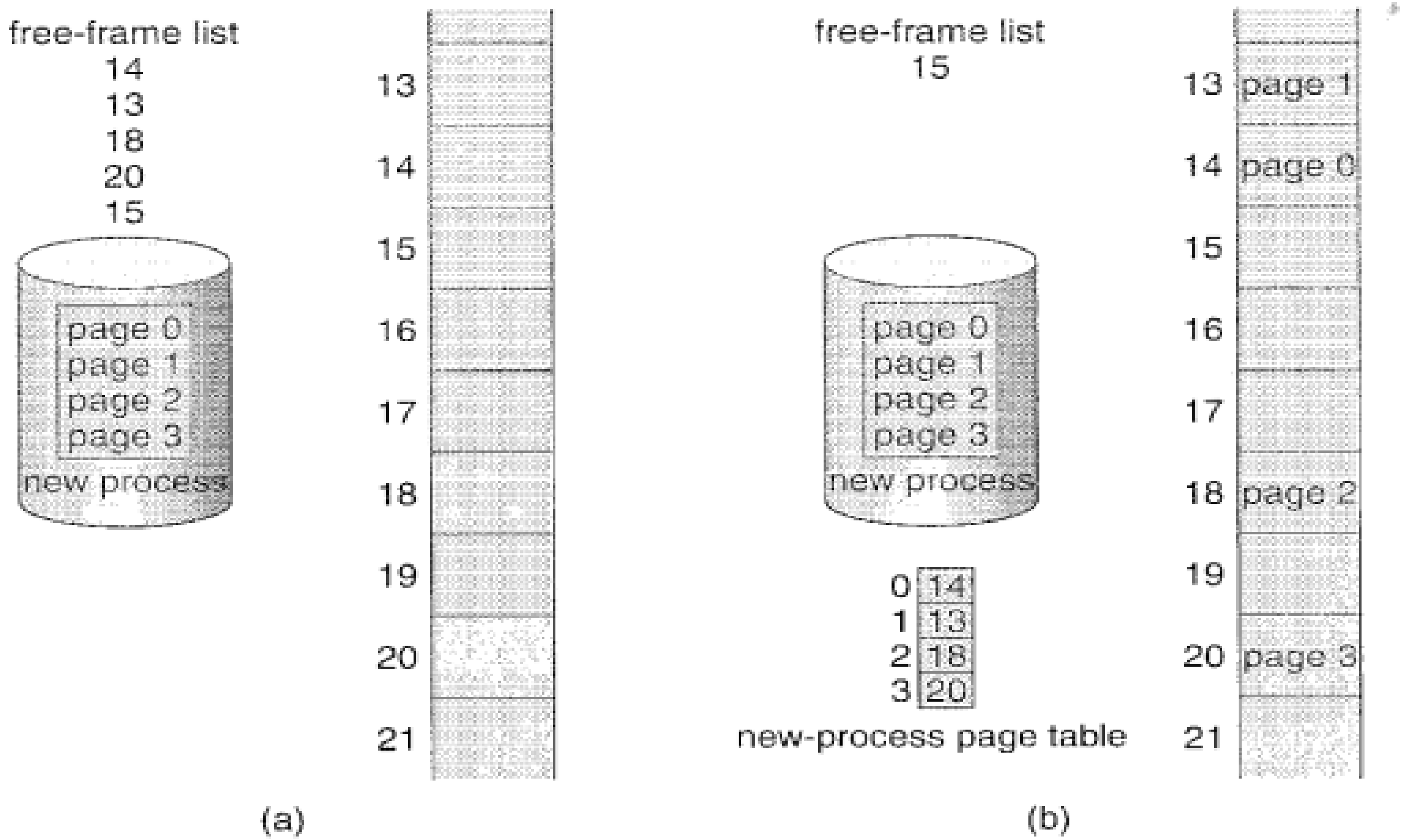
Logical pages are mapped to frames in the physical memory.

Paging



logical
address
physical
address
CPU
p
d
f
d
f0000 ... 0000
f1111 ... 1111
f
p
f
page table
physical
memory

Frame table

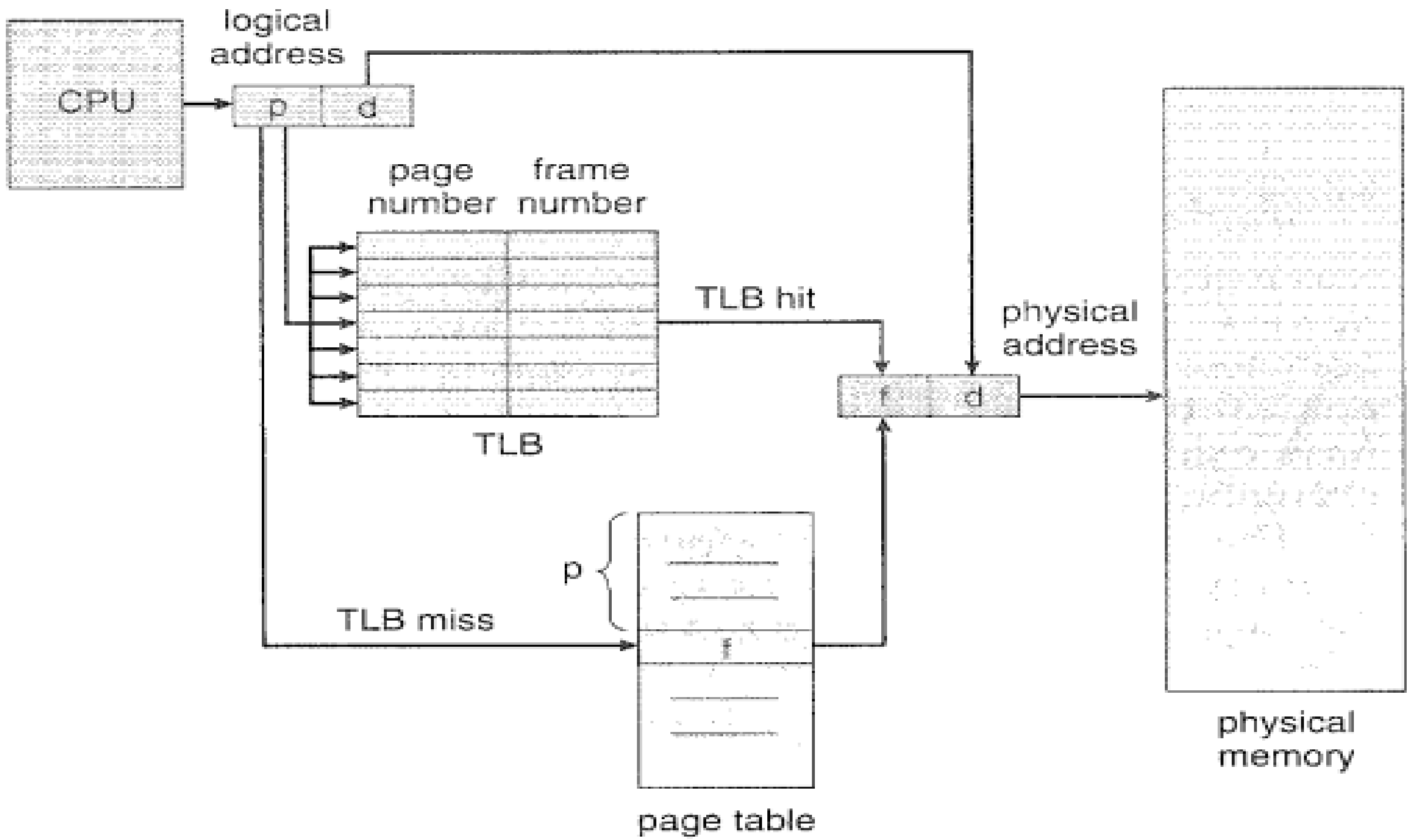


free-frame list
14
13
18
20
15
page 0
page 1
page 2
page 3
new process
13
14
15
16
17
18
19
20
21
(a)
free-frame list
15
page 0
page 1
page 2
page 3
new process
0 14
1 13
2 18
3 20
new-process page table
13 page 1
14 page 0
15
16
17
18 page 2
19
20 page 3
21
(b)

Hardware support for Paging

- ***Page table base register (PTBR)*** *– This contains the address of the page table. To change the page table only the entry in this register needs to be changed. It is done as a part of context switching.*
- *Translation Look aside Buffer (TLB)*

Transition Look aside Buffer (TLB)



Effective memory access time =
Percentage of TLB hit * time required to access the memory with TLB hit
+
percentage of TLB miss * time required to access memory with TLB miss

Search TLB = 20ns

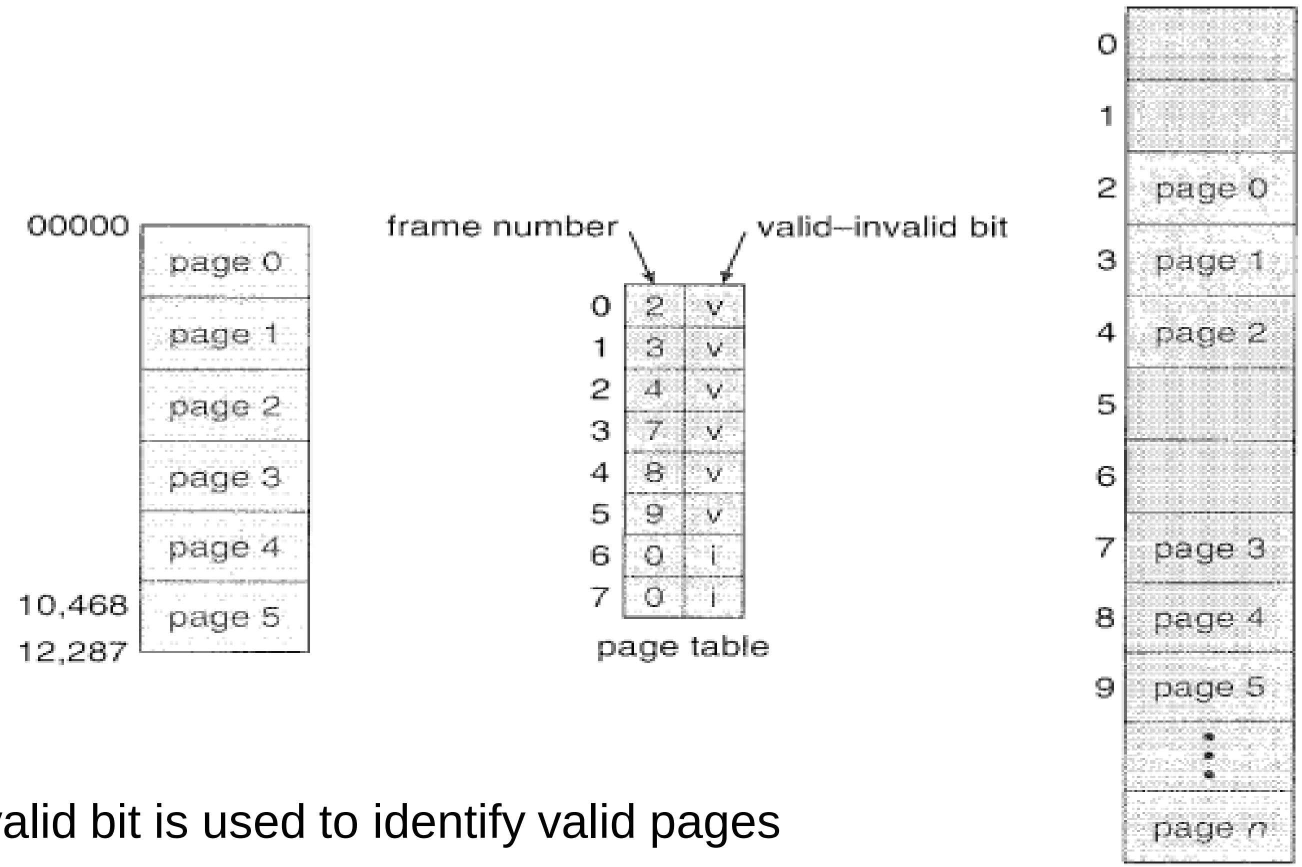
Accessing page table = 100 ns

Accessing memory = 100ns

TLB hit = 60%

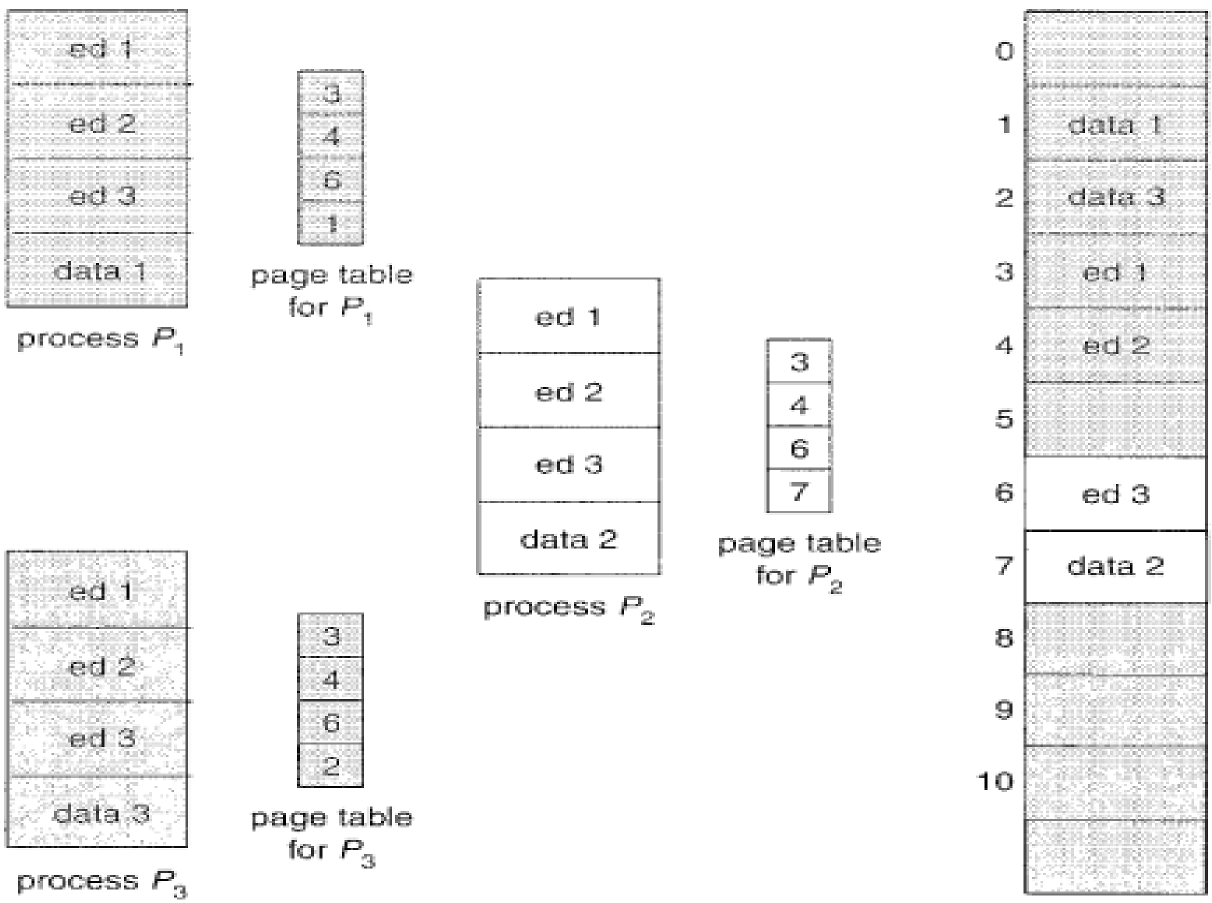
Find EMAT

Memory protection



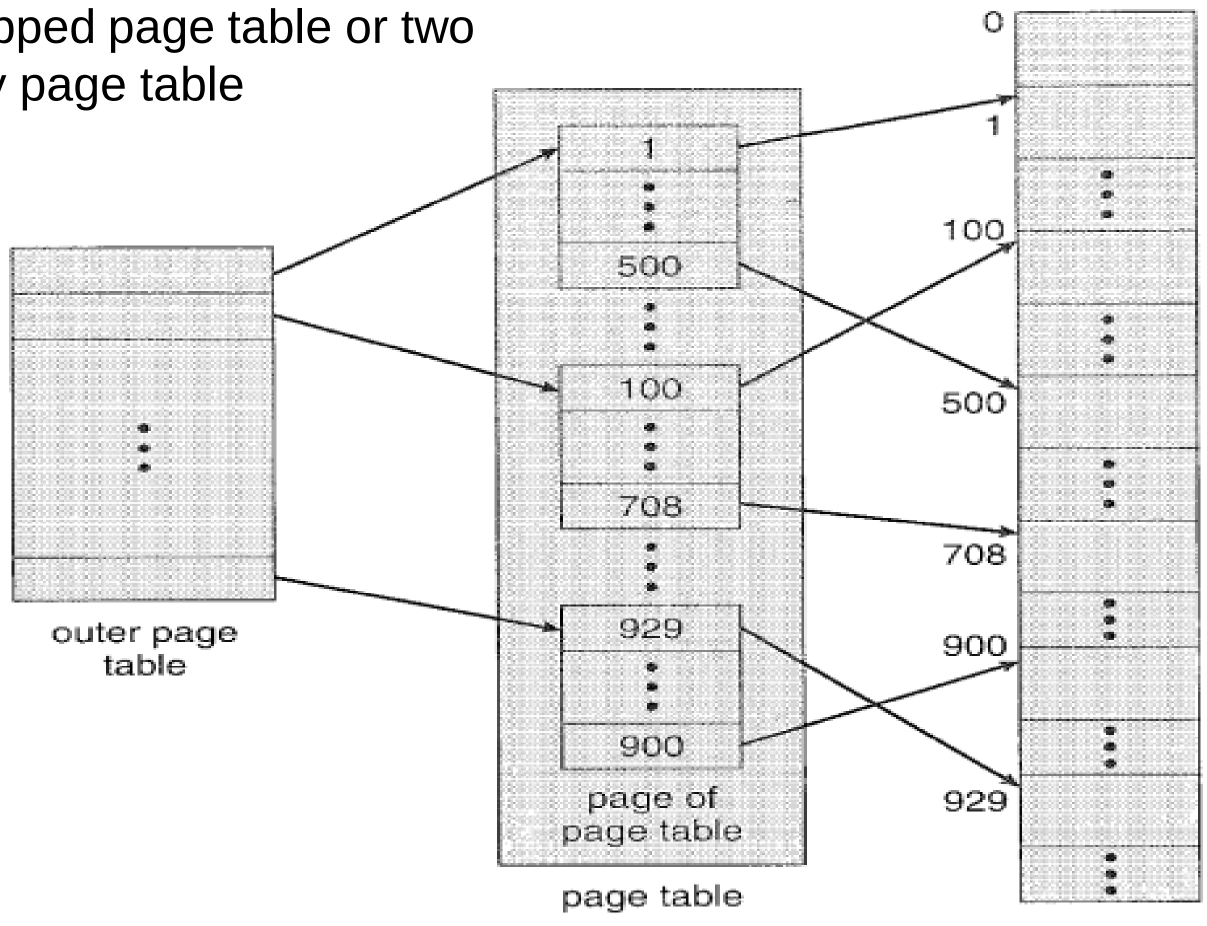
Valid /invalid bit is used to identify valid pages

Shared pages And Reentrant code



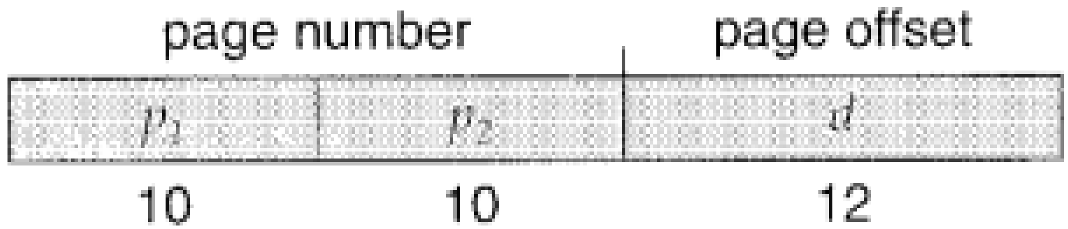
Page Table Structure

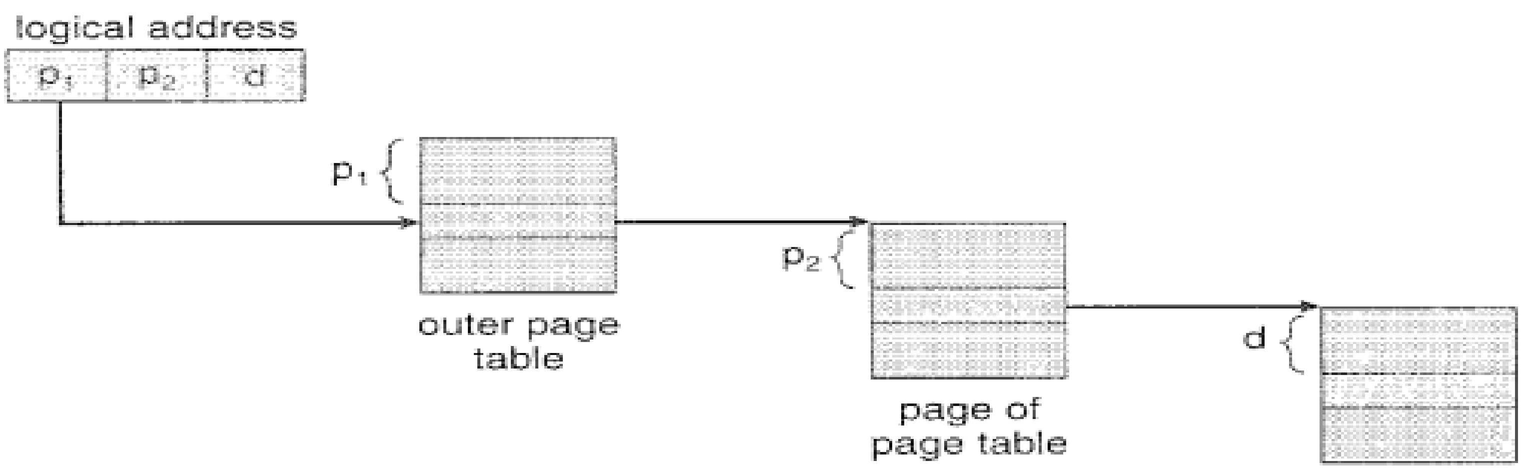
Also called as forward mapped page table or two way page table



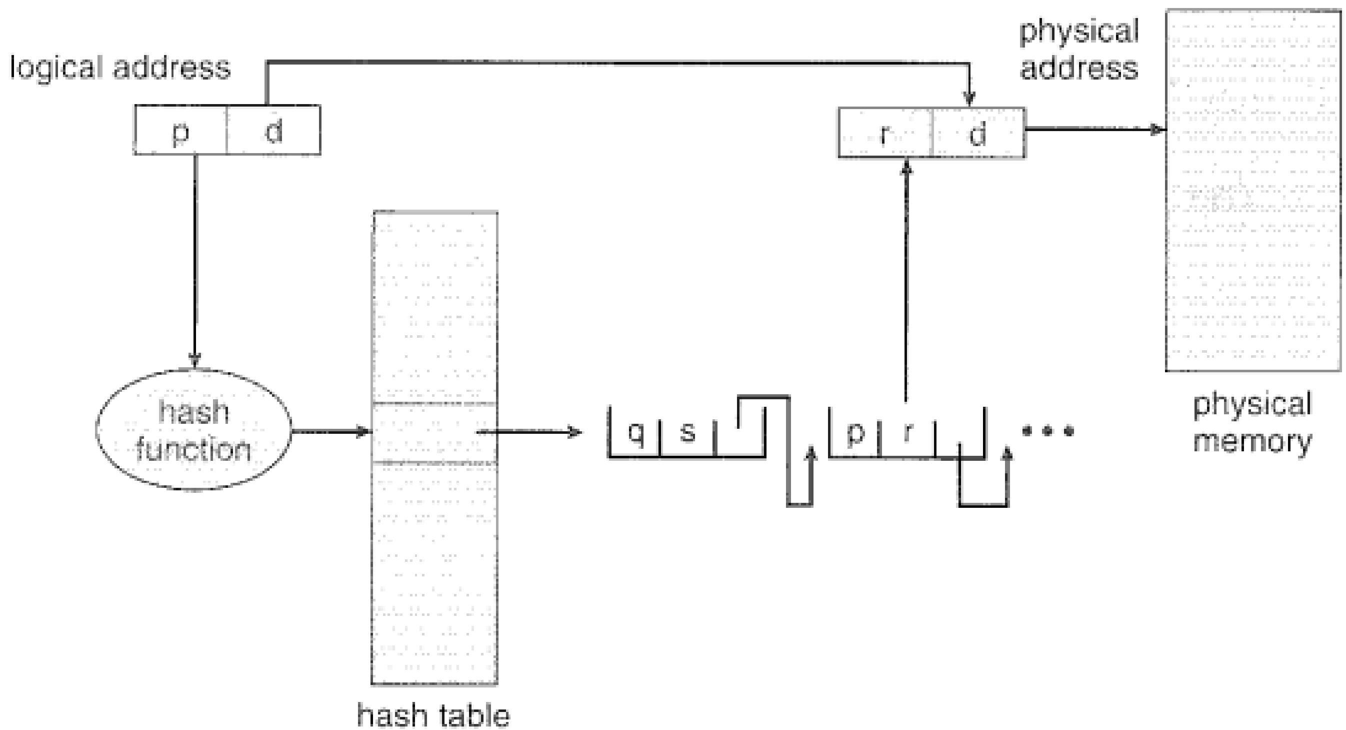
Page Table Structure

- *Example – 32 bit machine with page size of 4Kb*

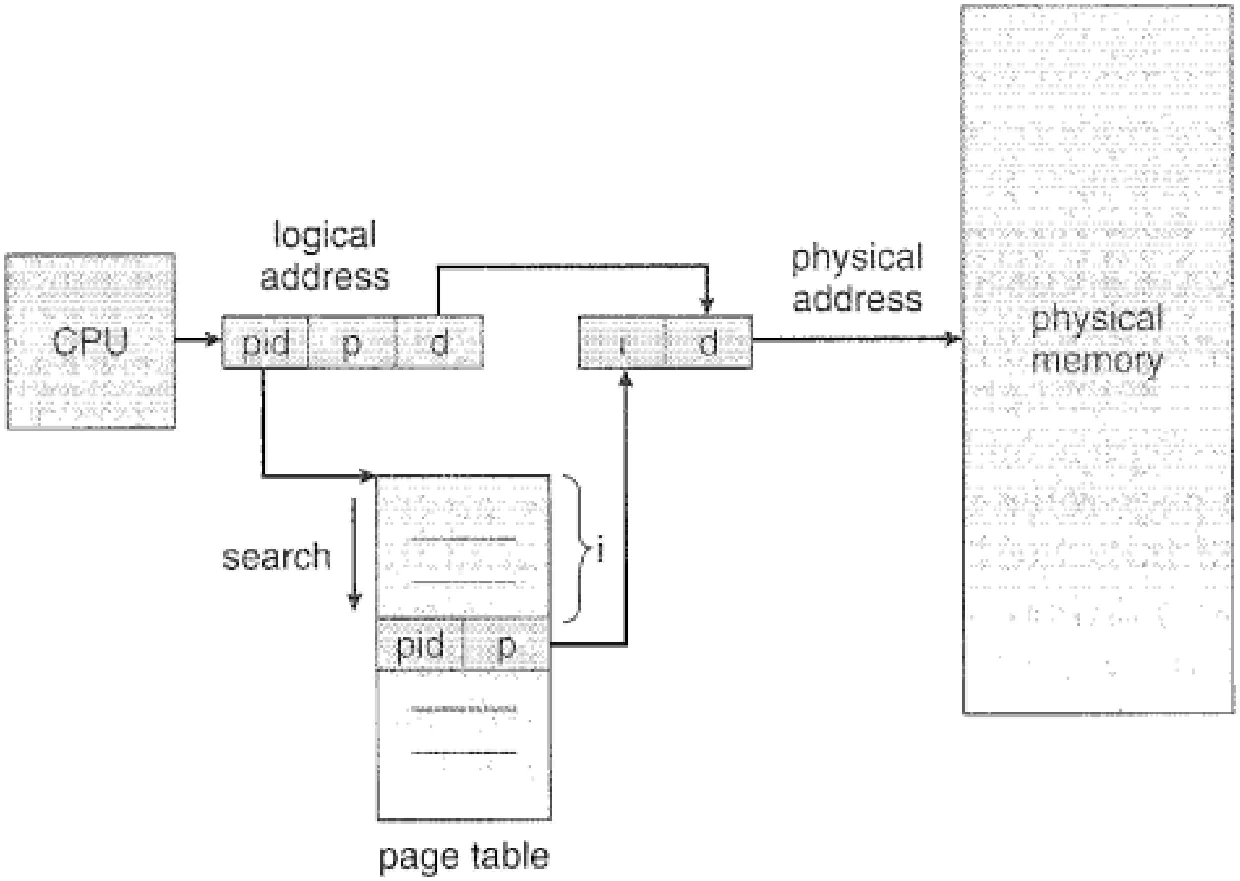




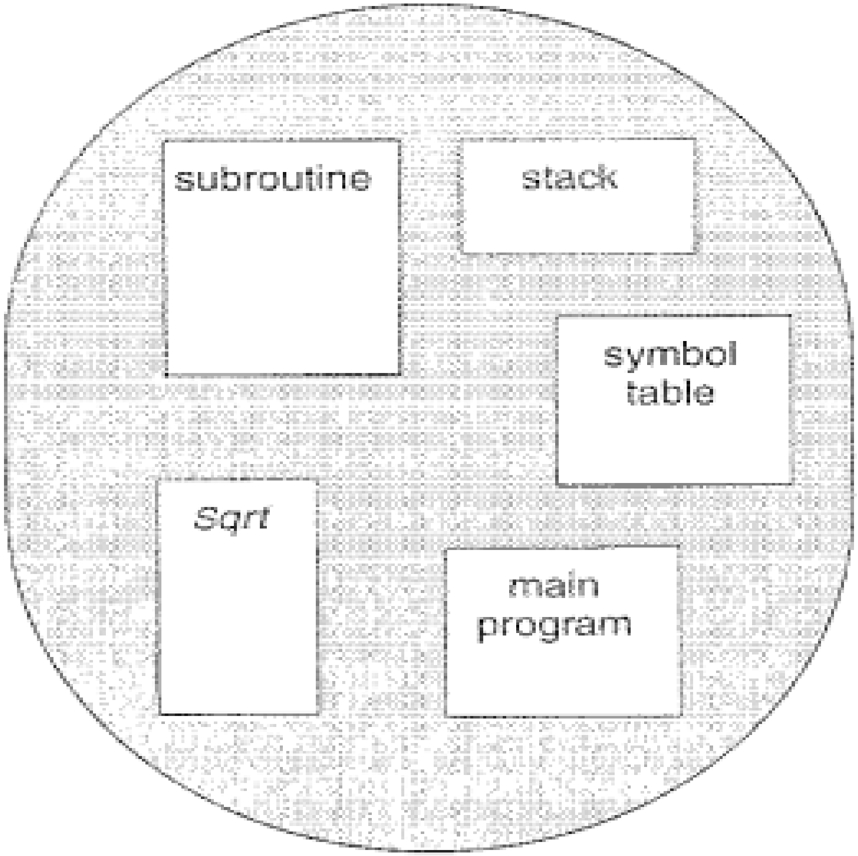
Hashed Page Table



Inverted Page Table



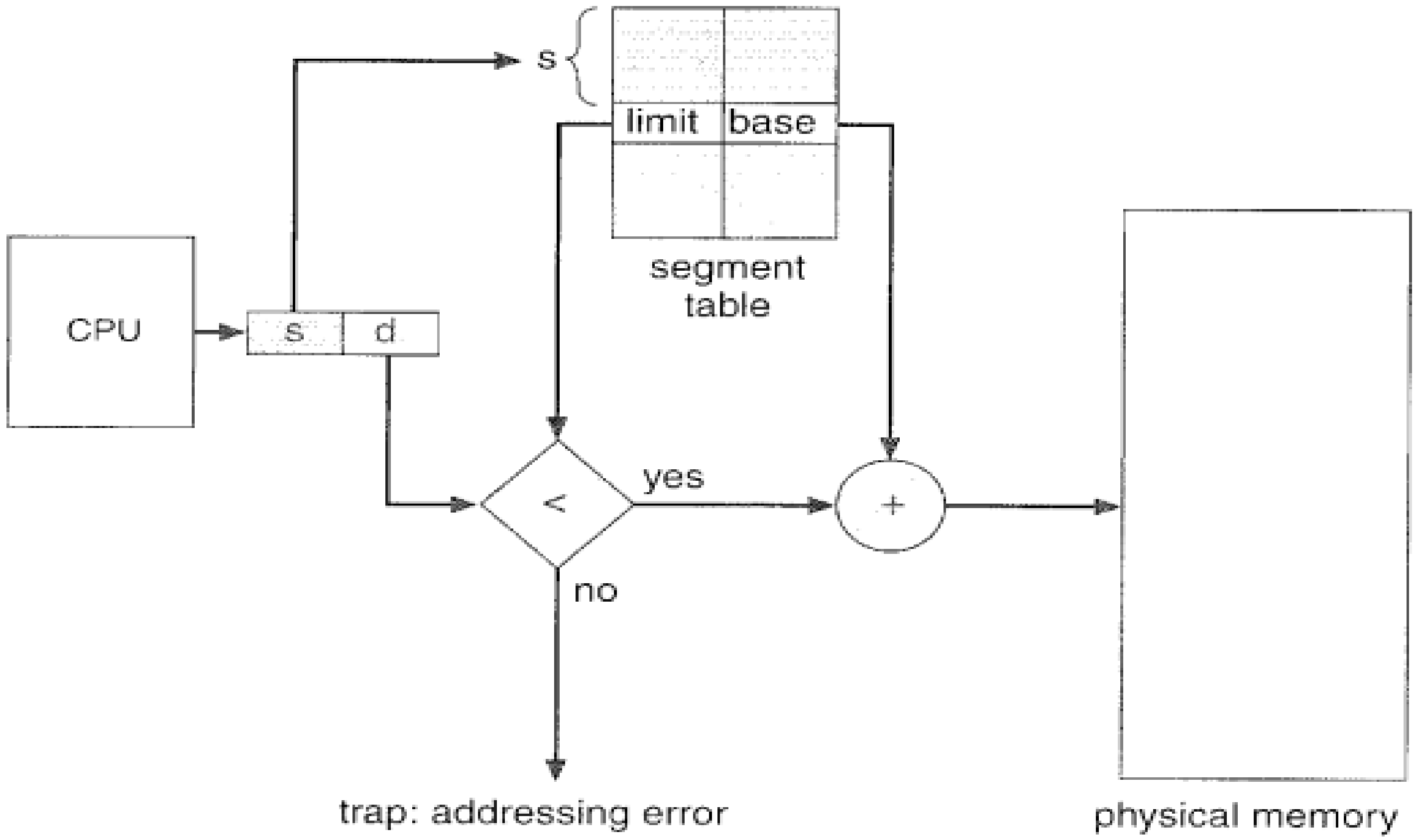
Segmentation

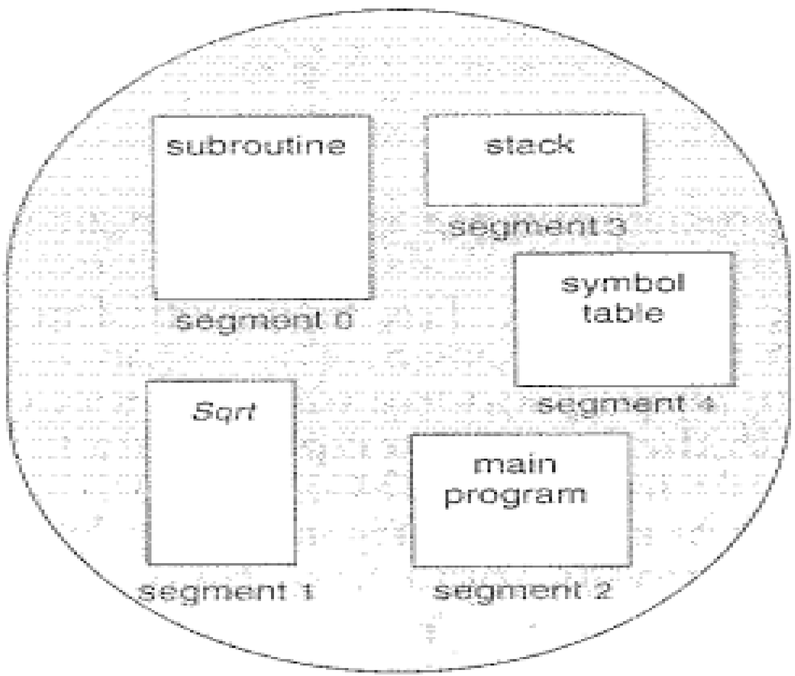


logical address

<segment-number, offset>.

Segmentation Hardware

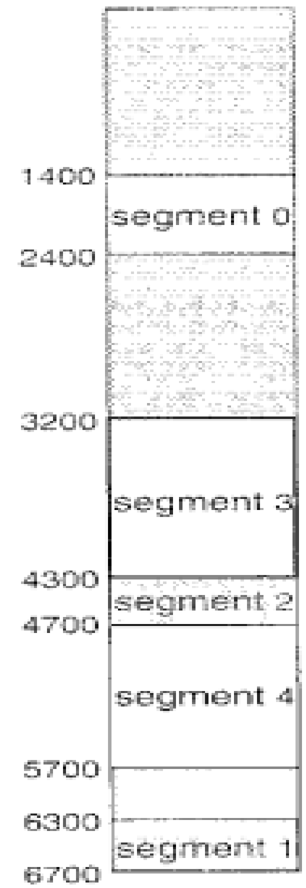




logical address space

	limit	base
0	1000	1400
1	400	6300
2	400	4300
3	1100	3200
4	1000	4700

segment table



physical memory

Address Binding

- ***Compile time*** – *if you know at compile time where will the process reside in the memory, an absolute address may be given by the compiler to every instruction;*

When the absolute address is not known at the compile time it provides the relocatable code during compilation.

- ***Load time*** *–Here the binding of the code with memory location is delayed to load time.*
- ***Linking Time or execution time***

Dynamic Loading and Linking

- *A routine is loaded only when it is called;*
- *All routines are kept in relocatable load format;*
- *Unused routine never loaded;*
- *Stubs*

- *Dynamic Link Libraries - system libraries that are linked at run time;*

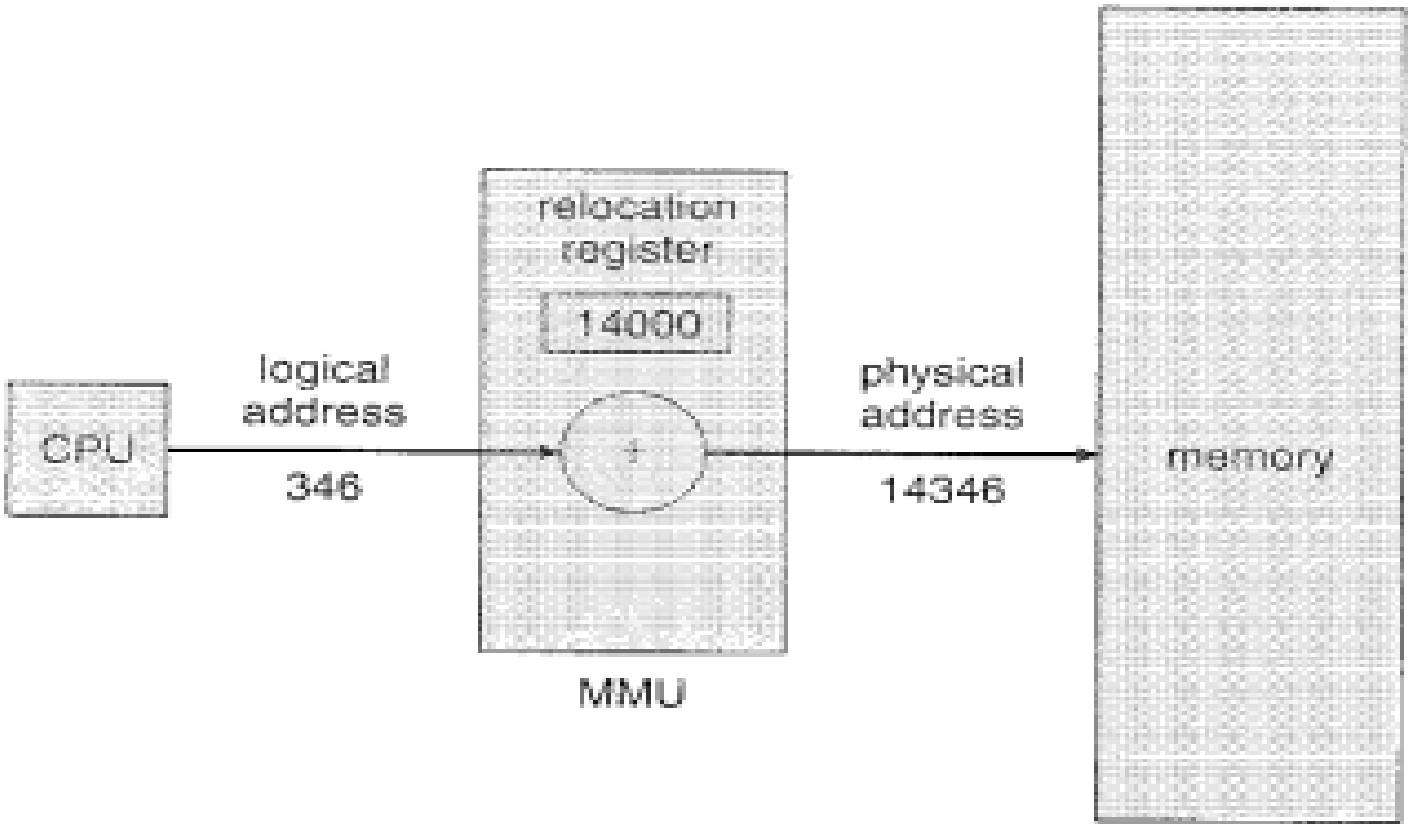
Logical Vs Physical Memory

- *Address generated by CPU*
- *Also called as Virtual address*

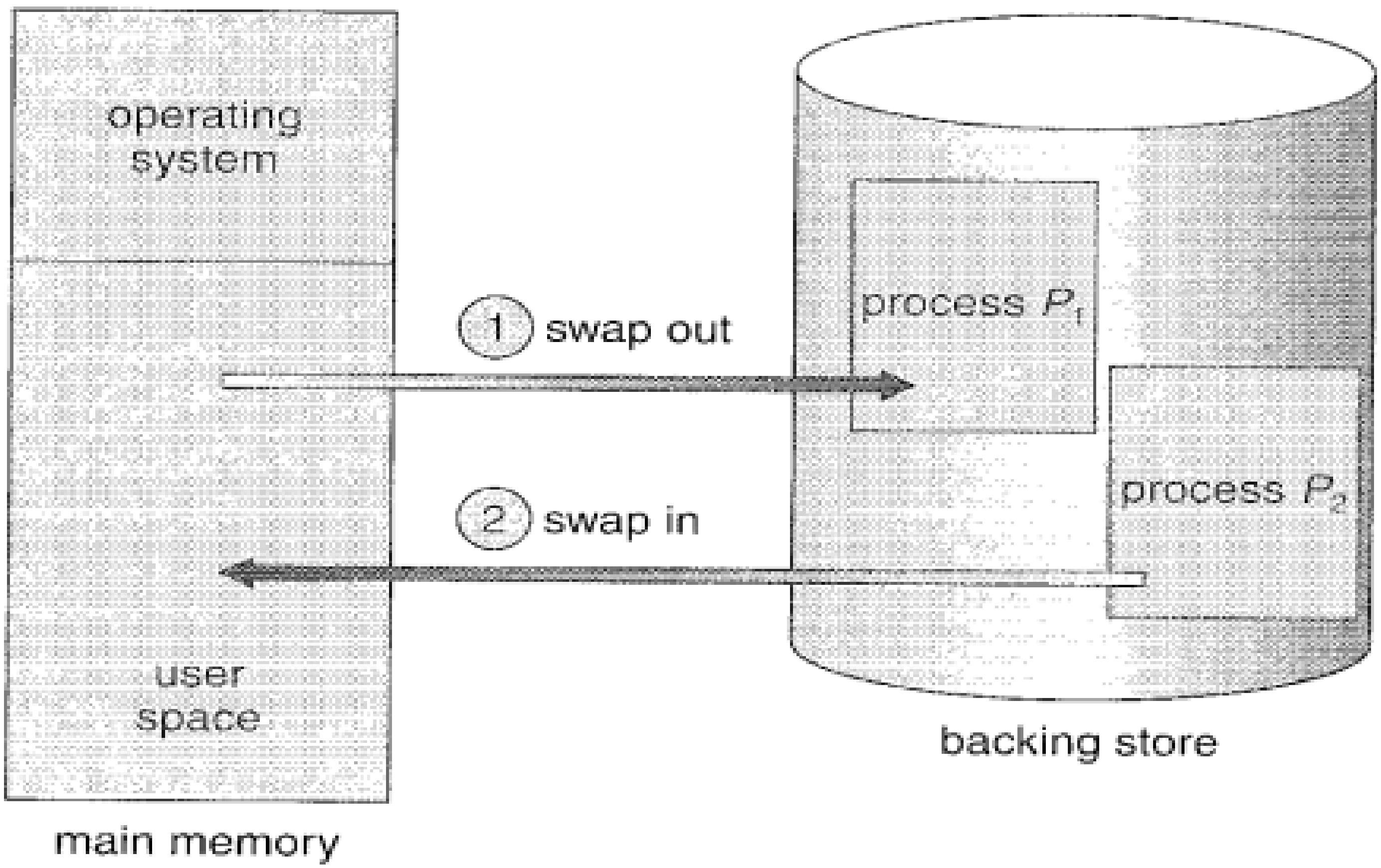
- *Address seen by Memory Management Unit (MMU)*
- *Loaded into Memory Address Register (MAR)*

Logical and Physical address is same for the address generated in compile and load time whereas they differ for link time binding.

Logical Address and Physical Address



Swapping


operating
system
user
space
main memory
1 swap out
2 swap in
process P_1
process P_2
backing store

Swapping

- *Context switching usually takes more time*

THANK YOU