
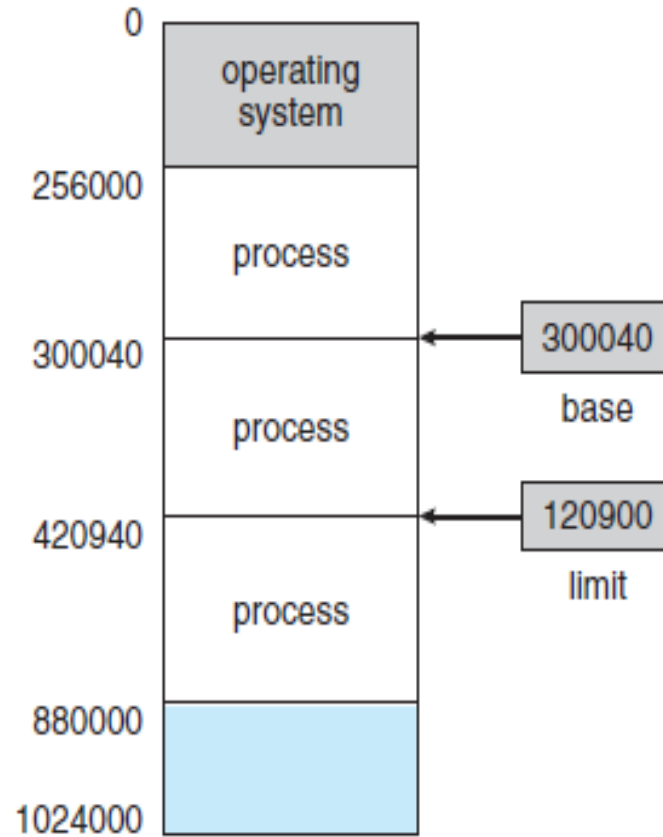


Memory Management

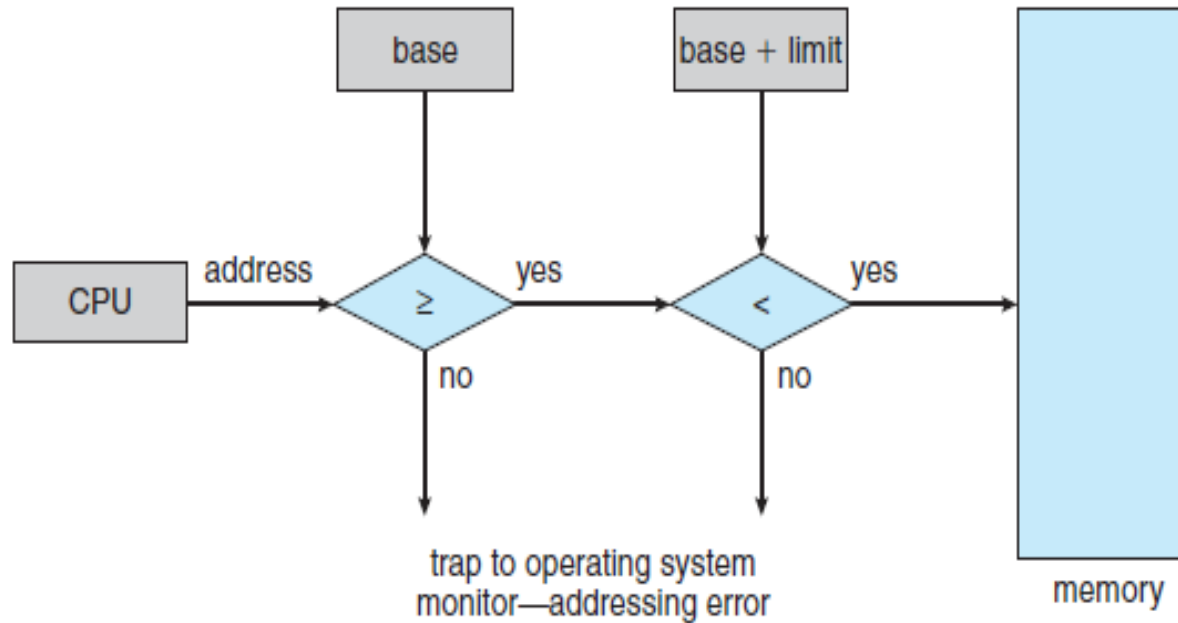
- 
- ❓ We begin our discussion by covering several issues that are pertinent to managing memory:
 - ❓ Basic hardware,
 - ❓ Binding of symbolic memory addresses to actual physical addresses, and the distinction between logical and physical addresses.

Basic Hardware

- ? Main memory
- ? Registers
- ? Cache
- ? Separate main memory for each process



Hardware address protection with base and limit registers



Address Binding

? binding of instructions and data to memory addresses

? **Compile time**

? **Load time**

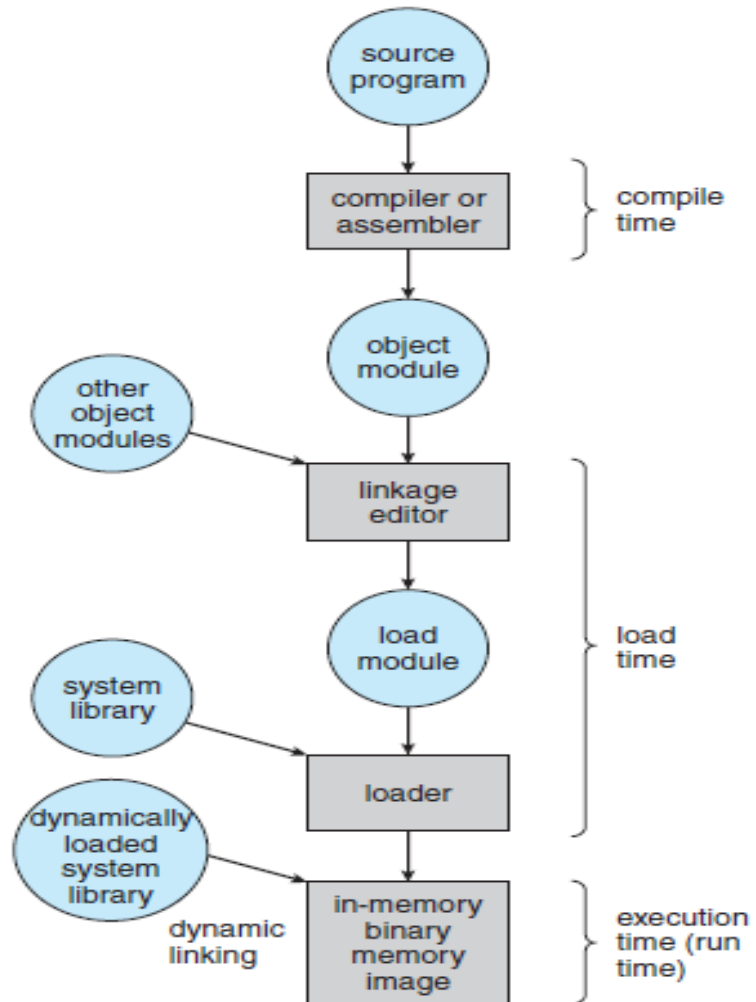
? **Execution time**

slido

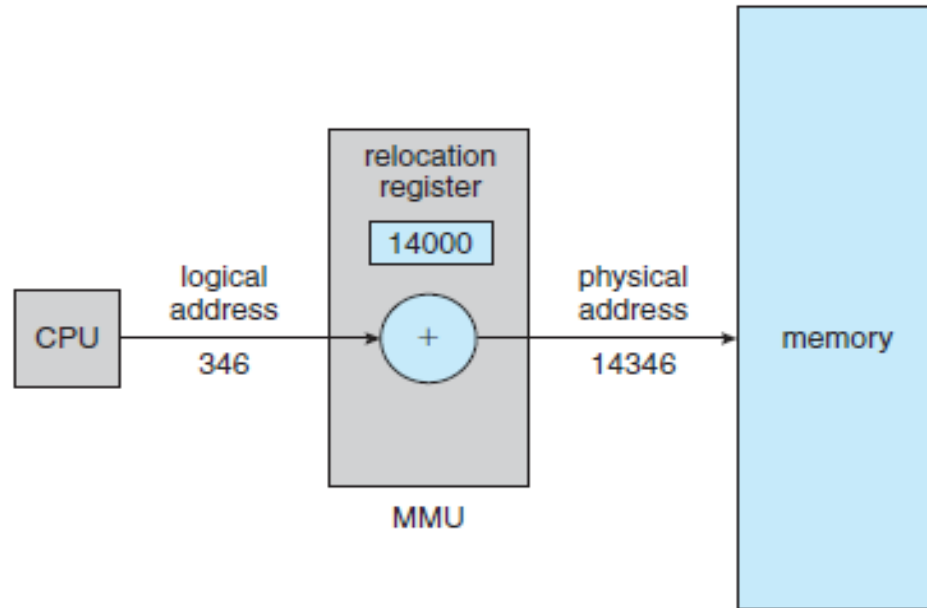


xyz???

ⓘ Start presenting to display the poll results on this slide.



Logical Versus Physical Address Space



Loading

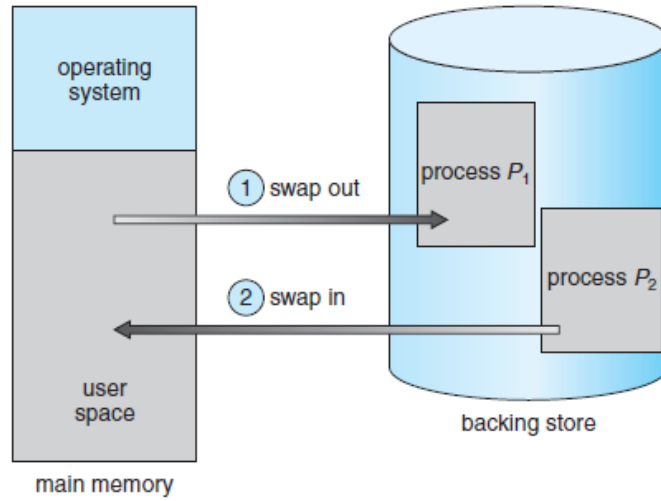
 Static

 Dynamic

Linking

- ❑ Static
- ❑ Dynamic

Swapping



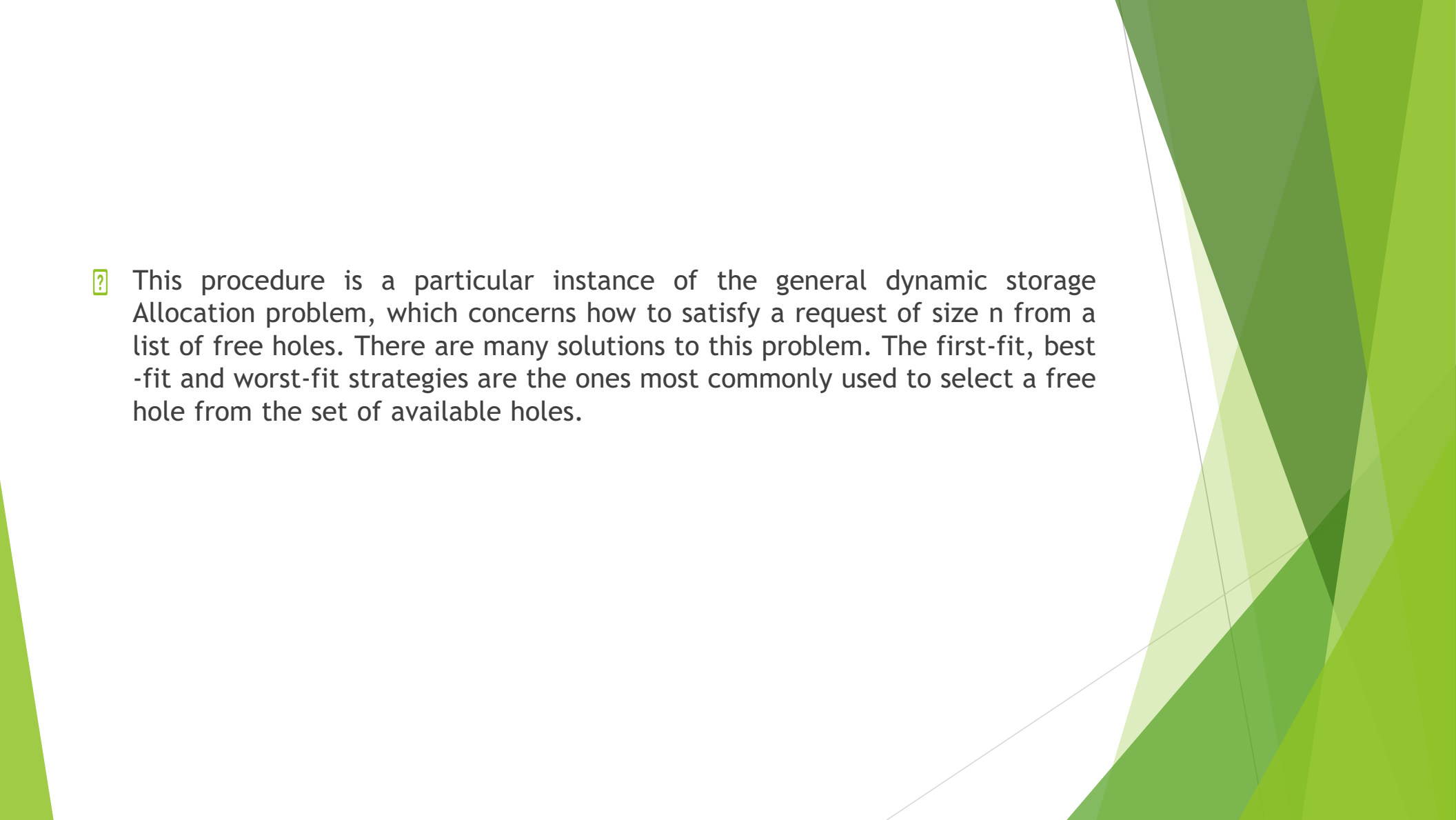
Contiguous Memory Allocation

- ❓ Memory Protection
- ❓ When the scheduler selects a process for execution, the dispatcher CPU loads the relocation and limit registers with the correct values as part of the context switch.

Memory Allocation

❓ fixed-sized partitions

❓ variable-partition

- 
- The background of the slide features abstract, overlapping green geometric shapes, primarily triangles and polygons, in various shades of green, creating a modern, layered effect on the right side.
- ❓ This procedure is a particular instance of the general dynamic storage Allocation problem, which concerns how to satisfy a request of size n from a list of free holes. There are many solutions to this problem. The first-fit, best-fit and worst-fit strategies are the ones most commonly used to select a free hole from the set of available holes.

Fragmentation

- ❓ External Fragmentation

- ❓ Internal Fragmentation

- ❓ One solution to the problem of external fragmentation is **compaction**. goal is to shuffle the memory contents so as to place all free memory together in one large block.

- ❓ Another possible solution to the external-fragmentation problem is to permit the logical address space of the processes to be non-contiguous, thus allowing a process to be allocated physical memory wherever such memory is available.

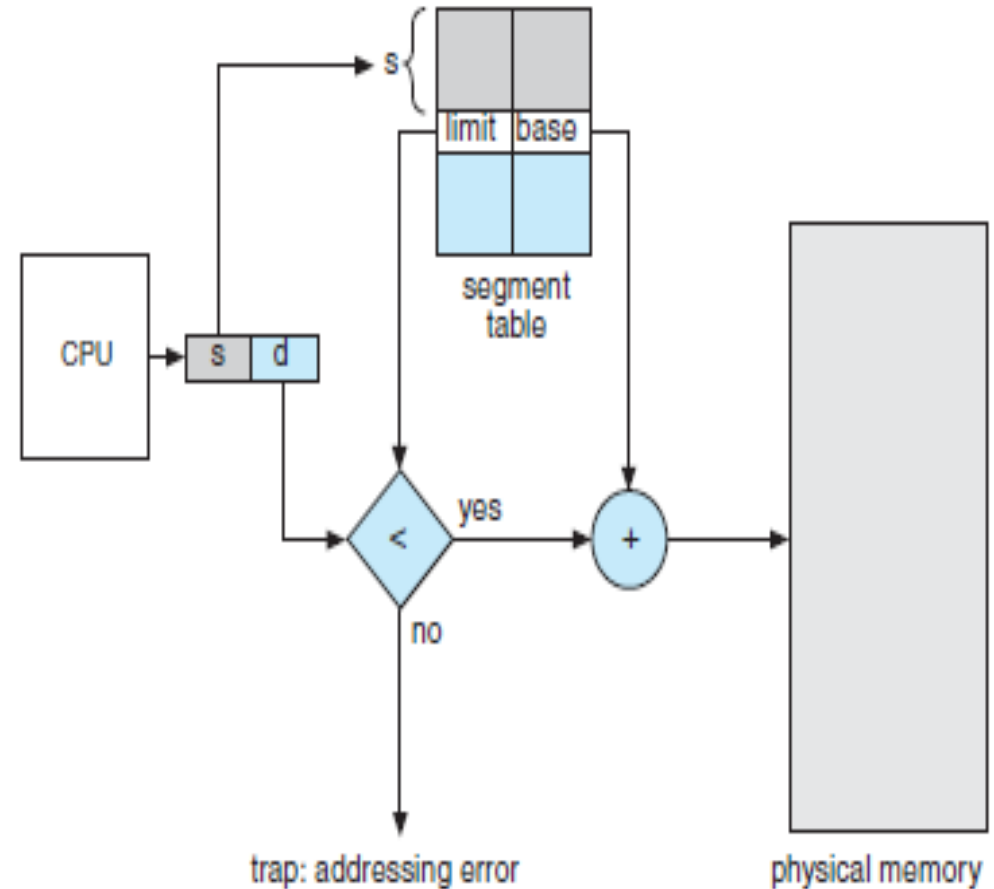
- ❓ Two complementary techniques achieve this solution: segmentation and paging.

Segmentation

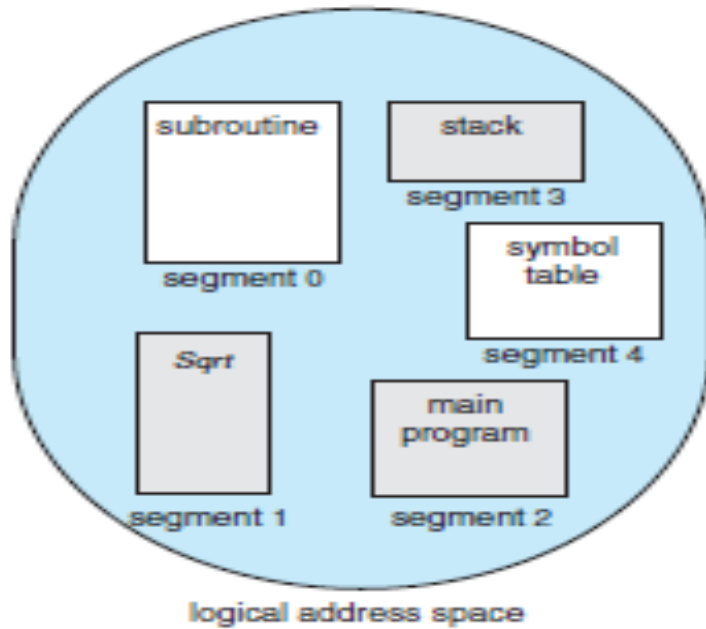
- ❓ A logical address space is a collection of segments.
- ❓ Each segment has a name and a length. The addresses specify both the segment name and the offset within the segment
- ❓ logical address consists of a two tuple:

`<segment-number, offset>`
- ❓ A C compiler might create separate segments for the following:
 1. The code
 2. Global variables
 3. The heap, from which memory is allocated
 4. The stacks used by each thread
 5. The standard C library

- Implementation to map two-dimensional user-defined addresses into one-dimensional physical addresses.
- This mapping is effected by a **segment table**. Each entry in the segment table has a **segment base** and a **segment limit**.
- A logical address consists of two parts: a segment number, s , and an offset into that segment, d .
- The segment number is used as an index to the segment table. The offset d of the logical address must be between 0 and the segment limit.
- When an offset is legal, it is added to the segment base to produce the address in physical memory of the desired byte. The segment table is thus essentially an array of base–limit register pairs.

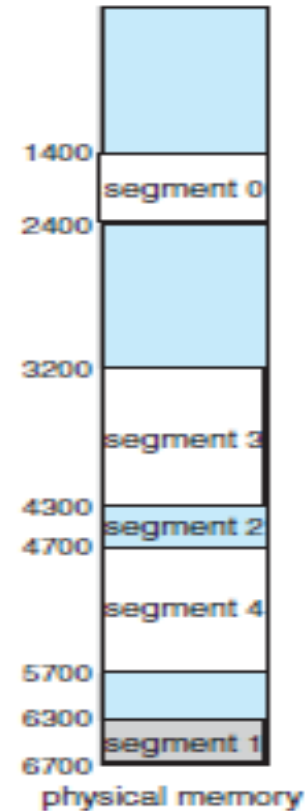


1. reference to byte 53 of segment 2
2. reference to byte 1222 of segment 0
3. reference to byte 853 of segment 3



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

segment table



Problems

Consider the following segment table-

Segment No.	Base	Length
0	1219	700
1	2300	14
2	90	100
3	1327	580
4	1952	96

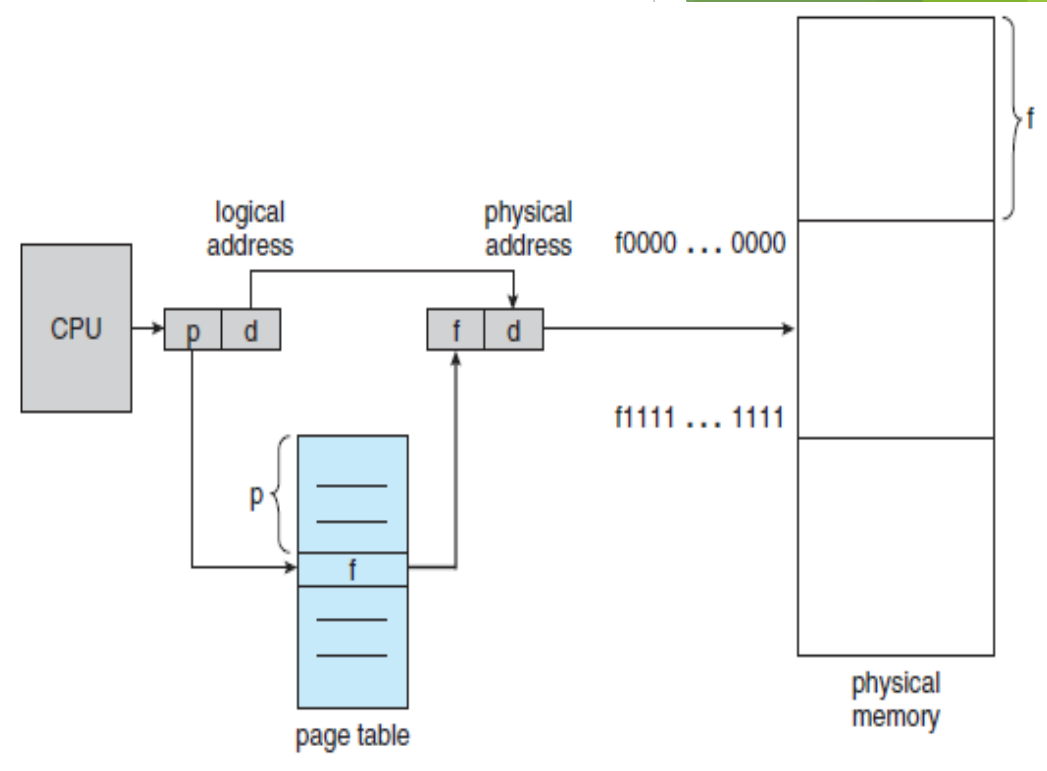
Which of the following logical address will produce trap addressing error?

1. 0, 430
2. 1, 11
3. 2, 100
4. 3, 425
5. 4, 95

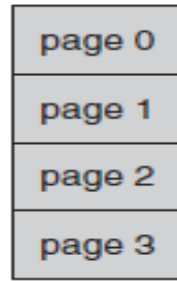
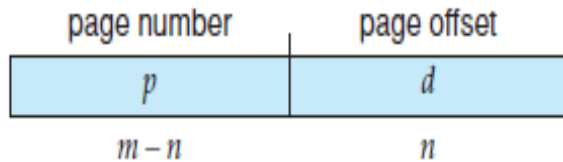
Calculate the physical address if no trap is produced.

Paging

- ? paging avoids external fragmentation
- ? The basic method for implementing paging involves breaking physical memory into fixed-sized blocks called frames and breaking logical memory into blocks of the same size called pages.
- ? Every address generated by the CPU is divided into two parts: a page number (p) and a page offset (d).
- ? The page number is used as an index into a page table.
- ? The page table contains the base address of each page in physical memory.



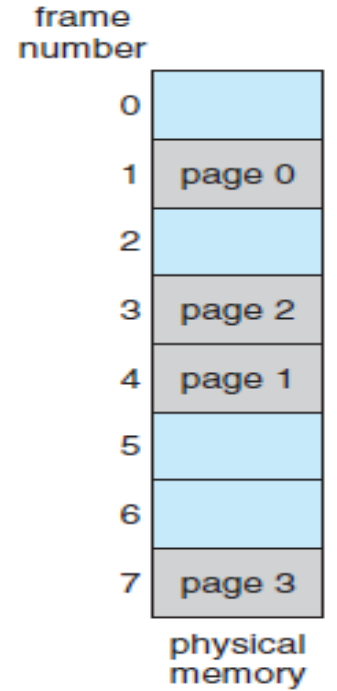
- ? the size of the logical address space is 2^m , and a page size is 2^n bytes,
- ? then the high-order $m-n$ bits of a logical address designate the page number, and the n low-order bits designate the page offset.



logical
memory

0	1
1	4
2	3
3	7

page table



physical
memory

- ? in the logical address, $n = 2$ and $m = 4$. Using a page size of 4 bytes and a physical memory of 32 bytes (8 pages),
- ? Logical address 0 is page 0, offset 0.
- ? logical address 0 maps to physical address 20 [= $(5 \times 4) + 0$].
- ? Logical address 3 (page 0, offset 3) maps to physical address 23 [= $(5 \times 4) + 3$].
- ? Logical address 13 maps to physical address 9.
- ? logical address 4 maps to physical address 24 [= $(6 \times 4) + 0$].

0	a
1	b
2	c
3	d
4	e
5	f
6	g
7	h
8	i
9	j
10	k
11	l
12	m
13	n
14	o
15	p

logical memory

0	5
1	6
2	1
3	2

page table

0	
4	i j k l
8	m n o p
12	
16	
20	a b c d
24	e f g h
28	

physical memory

- ❓ In the worst case, a process would need n pages plus 1 byte. It would be allocated $n + 1$ frames, resulting in internal fragmentation of almost an entire frame.
- ❓ The difference between the programmer's view of memory and the actual physical memory is reconciled by the address-translation hardware.
- ❓ Which frames are allocated, which frames are available, how many total frames there are, and so on. This information is generally kept in a data structure called a frame table.

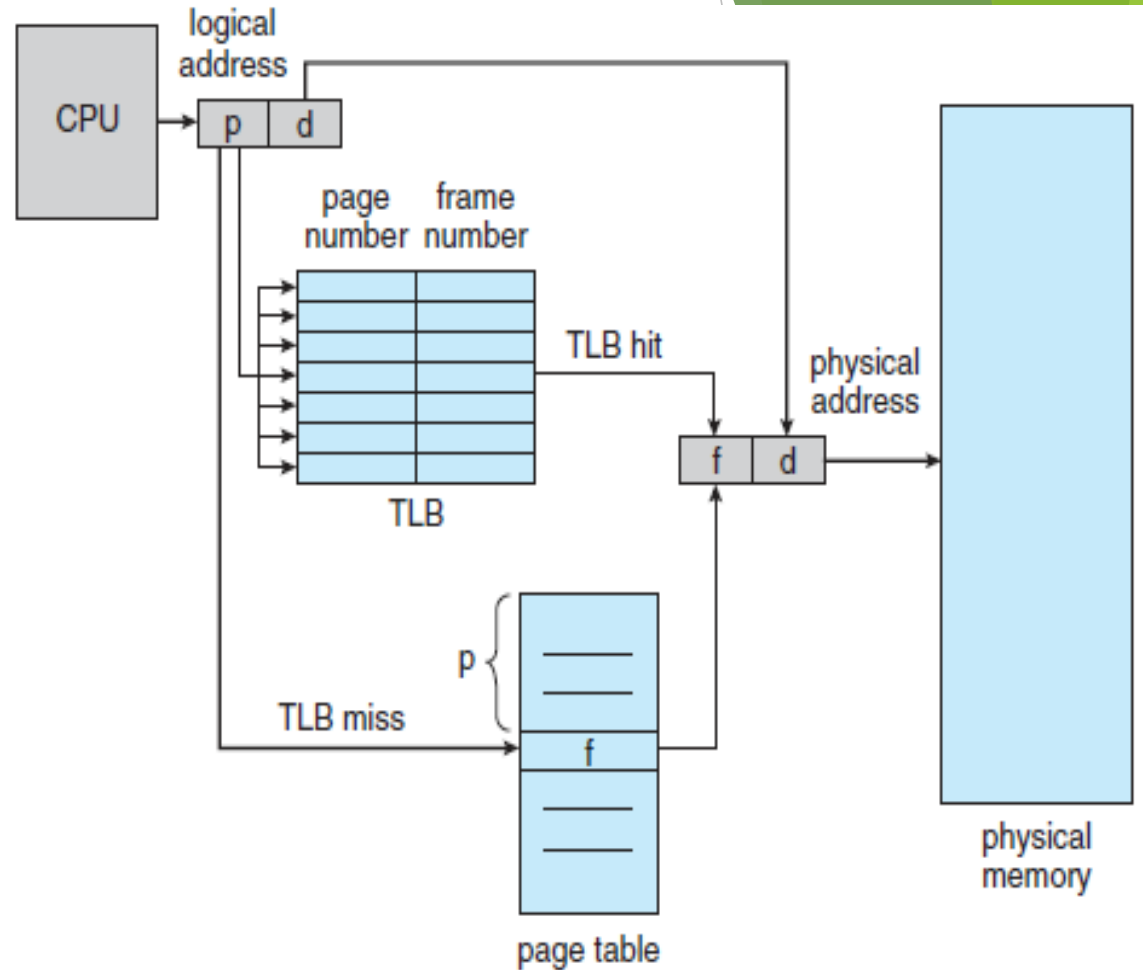
Problems

- ❓ Consider a system with LAS of 128MW and PA-24bits. The PAS is divided into 8K frames. What is page size and how many pages?
- ❓ Consider a single level paging scheme. The virtual address space is 4 MB and page size is 4 KB. What is the maximum page table entry size possible such that the entire page table fits well in one page?
- ❓ Consider a single level paging scheme. The virtual address space is 4 GB and page size is 128 KB. What is the maximum page table entry size possible such that the entire page table fits well in one page?

Hardware Support

- ? The page table is kept in main memory, and a page-table base register (PTBR) points to the page table
- ? If we want to access location i , we must first index into the page table, using the value in PTBR the offset by the page number for i - 1 MEMEORY ACCESS
- ? ACCESS desired place in memory -2 MEMEORY ACCESS
- ? Thus, memory access is slowed by a factor of 2.
- ? Solution– fast Lookup hardware cache called a translation look–aside buffer (TLB).
- ? The TLB contains only a few of the page-table entries. When a logical address is generated by the CPU, its page number is presented to the TLB. If the page number is found, its frame number is immediately available and is used to access memory.

- ? If the page number is not in the TLB (known as a TLB miss), a memory reference to the page table must be made.
- ? In addition, we add the page number and frame number to the TLB so that they will be found quickly on the next reference.
- ? If the TLB is already full of entries, an existing entry must be selected for replacement. Replacement policies range from least recently used (LRU) through round-robin to random.



- ? The percentage of times that the page number of interest is found in the TLB is called the hit ratio.
- ? The desired page number in the TLB 80 percent of the time. If it takes 100 nanoseconds to access memory, then a mapped-memory access takes 100 nanoseconds when the page number is in the TLB

effective access time = $0.80 * 100 + 0.20 * 200 == 120$ nanoseconds

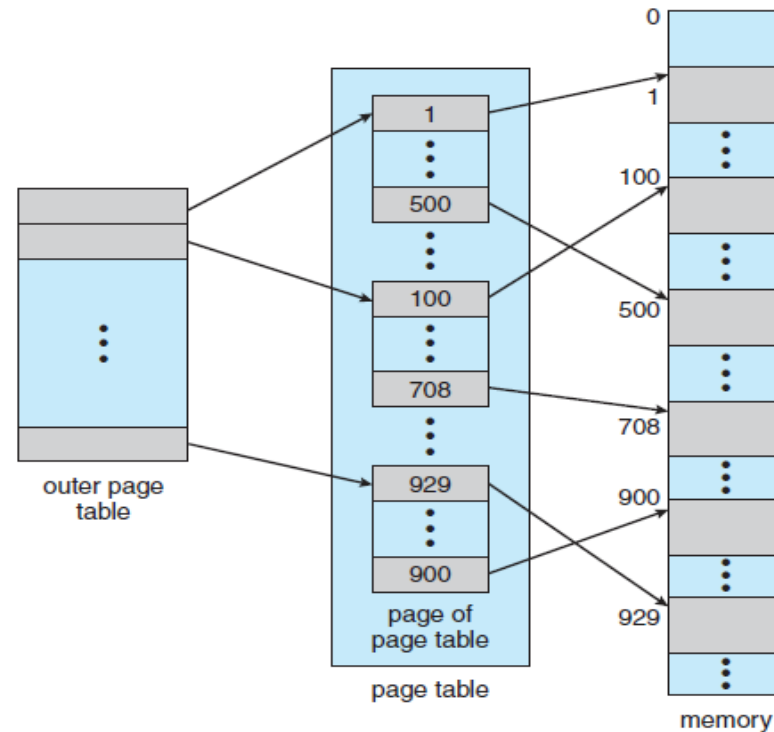
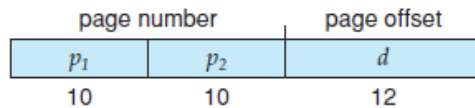
Problem

A paging scheme uses a Translation Lookaside buffer (TLB). A TLB access takes 10 ns and a main memory access takes 50 ns. What is the effective access time (in ns) if the TLB hit ratio is 90% and there is no page fault?

1. 54
2. 60
3. 65
4. 75

Structure of the Page Table: Hierarchical Paging

- ? In a 32-bit logical address space, if the page size in such a system is 4 KB (2^{12}), then a page table may consist of up to 1 million entries ($2^{32}/2^{12}=2^{20}$).



Problems

- ❓ Consider a system with multiple paging applicable. The page table is divided into 2k pages each of size of 4kw. If the PAS is 64mw which is divided into 16k frames and age table entry size is 2B. Then calculate
- ❓ length of logical address
- ❓ Length of physical address
- ❓ Outer Page Table Size
- ❓ Inner Page Table Size


Consider a system using multilevel paging scheme. The page size is 1 MB. The memory is byte addressable and virtual address is 64 bits long. The page table entry size is 4 bytes.

Find-

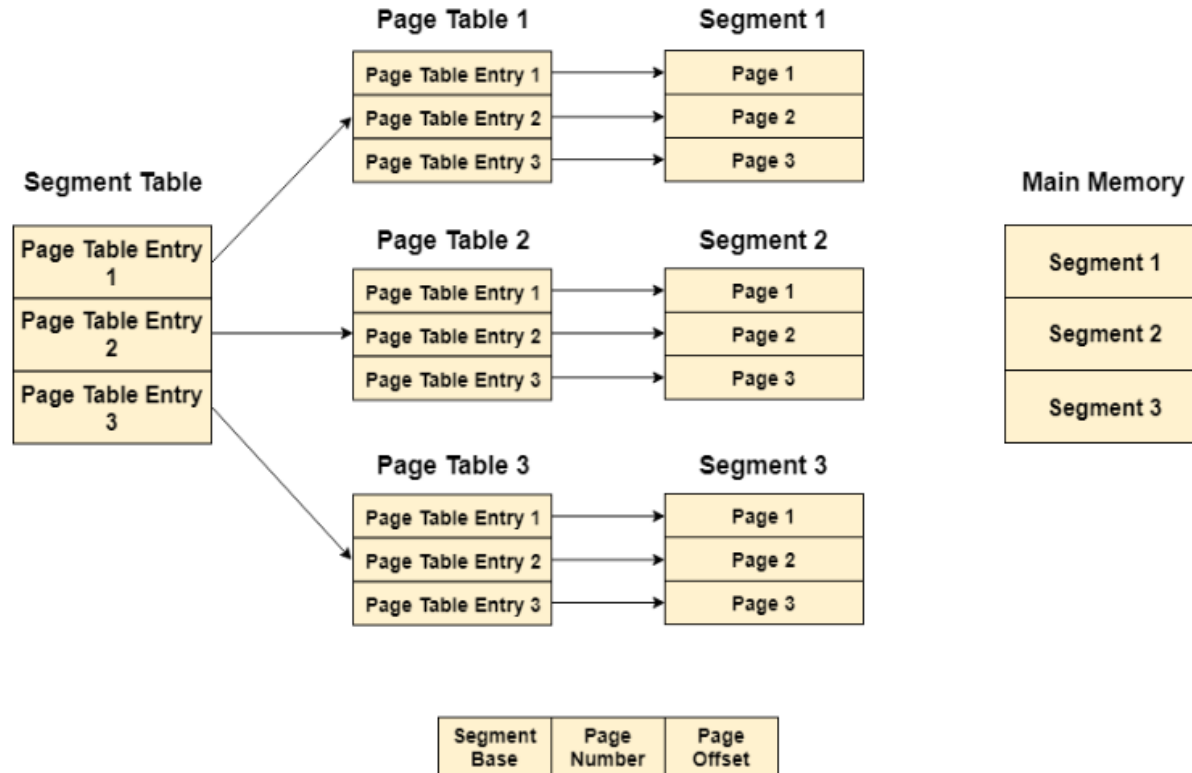
1. How many levels of page table will be required?
2. Give the divided physical address and virtual address.

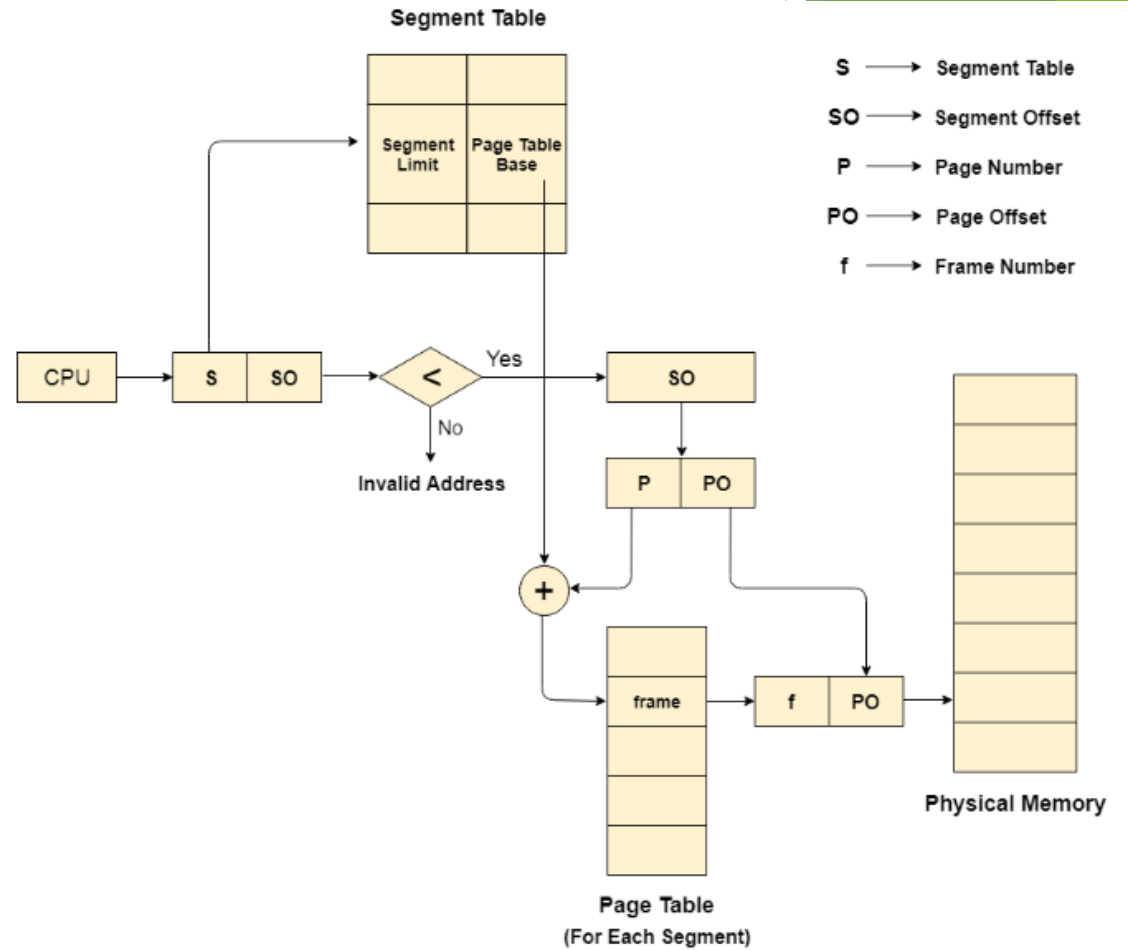
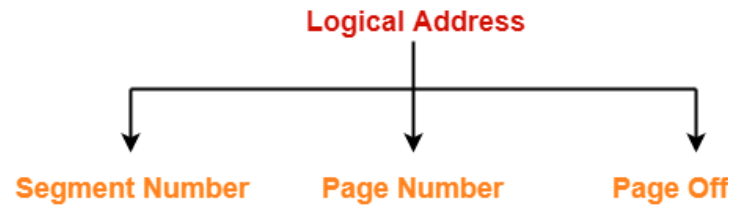
Inverted Page Tables

- ❓ Since, the table is sorted by virtual address, the operating system is able to calculate where in the table the associated physical address entry is located and to use that value directly.
- ❓ One of the drawbacks of this method is that each page table may consist of millions of entries.
- ❓ An inverted page table has one entry for each real page (or frame) of memory. Each entry consists of the virtual address of the page stored in that real memory location

- 
- ?
- Consider a system which has 34 bits of logical page size is 16KB and the memory is byte addressable. The page table entry size is 8Bytes. Then calculate:
- ?
- i) conventional page table size
- ?
- ii) Inverted page table size

Segmented paging





Problem

- ❓ A certain computer system has the segmented paging architecture for virtual memory. The memory is byte addressable. Both virtual and physical address spaces contain 2^{16} bytes each. The virtual address space is divided into 8 non-overlapping equal size segments. The memory management unit (MMU) has a hardware segment table, each entry of which contains the physical address of the page table for the segment. Page tables are stored in the main memory and consists of 2 byte page table entries. What is the minimum page size in bytes so that the page table for a segment requires at most one page to store it?