For The following Exercises, mark the answers True and False

1. An operating system is an example of application software. **False**
2. An operating system provides a basic user interface that allows the user to use the computer. **True**
3. A computer can have more than one operating system, but only one OS is in control at any given time. **True**
4. Multiprogramming is the technique of using multiple CPUs to run programs. **False**
5. Batch processing implies a high level of interaction between the user and the program. **False**
6. A timesharing system allows multiple users to interact with a computer at the same time. **True**
7. A logical address specifies an actual location in main memory. **False**
8. An address in a single contiguous memory management system is made up of a page and an offset. **False**
9. The bounds register contains the last address of a partition. **False** It contains the length of the partition.
10. The first page in a paged memory system is page 0. **True**
11. A process in the running state is currently being executed by the CPU. **True**
12. The process control block (PCB) is a data structure that stores all information about a process. **True**
13. CPU scheduling determines which programs are in memory. **False**
14. The first-com, first-served scheduling algorithm is provably optimal. **False**
15. A time slice is the amount of time each process is given before being preempted in a round robin scheduler. **True**
For The following Exercises, match the following software type with its definition.

A. Systems software
B. Operating system
C. Application software

24. Programs that help us solve real-world problems.  **Application software**
25. Programs that manage a computer system and interact with hardware.  **Systems software**
26. Programs that manage computer resources and provide an interfaces for other programs.  **Operating system**

For The following Exercises are problems or short answer questions.

27. Distinguish between application software and system software.
   Systems software are tools to help others write programs; they manage a computer system and interact with hardware. Application software are programs to solve specific problems.

28. What is an operating system?
   An operating system is a piece of software that manages a computer's resources and provides an interface for system interaction.

29. Explain the term *multiprogramming*.
   Multiprogramming is the technique of keeping multiple programs in main memory at the same time, each competing for time on the CPU.

30. The following terms relate to how the operating system manages multiprogramming. Describe the part each plays in this process.
   A. Process
   A process is a program in execution.
   B. Process management
   Process management is keeping track of necessary information for active processes.
   C. Memory management
   Memory management is keeping track of how and where programs are loaded into main memory.
   D. CPU scheduling
   CPU scheduling is determining which process in memory is given access to the CPU so that it may execute.

31. Define *timesharing*.
   Timesharing is a technique by which CPU time is shared among multiple interactive users at the same times.
32. What is the relationship between multiprogramming and timesharing?
Multiprogramming allows multiple processes to be active at once.
Timesharing allows the multiple processes to be interactive ones.

33. Why do we say that users in a timesharing system have their own virtual machine?
Users have the illusion of having the computer all to themselves.

34. a virtual machine as a hypothetical machine designed to illustrate important features of a real machine. In this chapter, we define a virtual machine as the illusion created by a timesharing system that each user has a dedicated machine. Relate these two definitions.
The illusion created in a timesharing situation is that the user owns a single hypothetical machine. The hypothetical machine illustrates the important features of the single machine the user needs.

35. How does the timesharing concept work?
Each user is represented by a login process that runs on the mainframe. When the user runs a program, another process is created that competes for CPU time with other processes. The rationale is that the computer is so fast that it can handle multiple users without anyone having to wait.

36. What is a real-time system?
A real-time system is a system in which the speed of an answer is crucial.

37. What is response time?
Response time is how long it takes to get an answer. The expression comes from the delay between receiving a stimulus (asking a question) and producing a response (answering the question).

38. What is the relationship between real-time systems and response time?
Time is critical in many real-time situations, so the response time must be kept to a minimum.

39. In a multiprogramming environment, many processes may be active. What are the tasks that the OS must accomplish in order to manage the memory requirements of active processes?
The OS must keep track of where and how a program resides in memory and convert logical program addresses into actual memory addresses.

40. Distinguish between logical addresses and physical addresses.
A physical address is an actual address in the computer's main memory device. A logical address is an address relative to the program. A logical address is sometimes called a relative address for obvious reasons.

41. What is address binding?
Address binding is the mapping of a logical address into a physical address.
42. Name three memory management techniques and give the general approach taken in each.

**Single Task memory management:**
Only the OS and one application program are loaded into memory at the same time.

**Multiple Task memory management:**
More than one program is loaded into memory with the OS at the same time. Each application program is given its own partition of memory.

**Paging:**
Main memory is divided into fixed-sized blocks called frames and processes are divided into fixed-sized blocks called pages. Any number of programs can be loaded with the OS, but a process does not necessarily have to be in contiguous memory and not all of a process need be in memory at the same time.

43. When is a logical address assigned to a variable?
When the program is compiled.

44. When does address binding occur?
When the program is loaded into memory.

45. How is memory divided in the single contiguous memory management approach?
Memory is divided into two sections, one for the operating system and one for the application program.

46. When a program is compiled, where is it assumed that the program will be loaded into memory? That is, where are logical addresses assumed to begin?
At location 0.

47. If, in a single contiguous memory management system, the program is loaded at address 30215, compute the physical addresses (in decimal) that correspond to the following logical addresses:

   A. 9223  39438
   B. 2302  32517
   C. 7044  37259

48. In a single contiguous memory management approach, if the logical address of a variable is \( L \) and the beginning of the application program is \( A \), what is the formula for binding the logical address to the physical address?
\[ L + A \]

49. If, in a fixed partition memory management system, the current value of the base register is 42993 and the current value of the bounds register is 2031, compute the physical addresses that correspond to the following logical addresses:

   A. 104  43097
   B. 1755 44748
50. If more than one partition is being used (either fixed or dynamic), what does the base register contain?

The base register contains the beginning address of the current partition.

51. Why is the logical address compared to the bounds register before a physical address is calculated?

The bounds register contains the length of the current partition. If the logical address is greater than the bounds register, then the physical address is not within the current partition.

52. If, in a dynamic partition memory management system, the current value of the base register is 42993 and the current value of the bounds register is 2031, compute the physical addresses that correspond to the following logical addresses:

A. 104 43097
B. 1755 44748
C. 3041 Address out of bounds of partition.

Exercises 55 and 56 use the following state of memory.

<table>
<thead>
<tr>
<th>Operating System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process 1</td>
</tr>
<tr>
<td>New process</td>
</tr>
<tr>
<td>Process 2</td>
</tr>
<tr>
<td>Process 3</td>
</tr>
<tr>
<td>Empty 52 blocks</td>
</tr>
<tr>
<td>Empty 100 blocks</td>
</tr>
</tbody>
</table>

55. If the partitions are fixed and a new job arrives requiring 52 blocks of main memory, show memory after using each of the following partition selection approaches:

A. First fit  B. Best fit  C. Worst fit

A. First fit  B. Best fit  C. Worst fit
If the partitions are dynamic and a new job arrives requiring 52 blocks of main memory, show memory after using each of the following partition selection approaches:

A. First fit  
B. Best fit  
C. Worst fit

<table>
<thead>
<tr>
<th>A. First fit</th>
<th>B. Best fit</th>
<th>C. Worst fit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating System</td>
<td>Operating System</td>
<td>Operating System</td>
</tr>
<tr>
<td>Process 1</td>
<td>Process 1</td>
<td>Process 1</td>
</tr>
<tr>
<td>New process</td>
<td>Empty 60 blocks</td>
<td>Empty 60 blocks</td>
</tr>
<tr>
<td>Process 2</td>
<td>Process 2</td>
<td>Process 2</td>
</tr>
<tr>
<td>Process 3</td>
<td>Process 3</td>
<td>Process 3</td>
</tr>
<tr>
<td>Empty 52 blocks</td>
<td>New process</td>
<td>Empty 52 blocks</td>
</tr>
<tr>
<td>Empty 100 blocks</td>
<td>Empty 100 blocks</td>
<td>New process</td>
</tr>
<tr>
<td>Empty 6 blocks</td>
<td>Empty 100 blocks</td>
<td>Empty 48 blocks</td>
</tr>
</tbody>
</table>

If the partitions are dynamic and a new job arrives requiring 52 blocks of main memory, show memory after using each of the following partition selection approaches:

A. First fit  
B. Best fit  
C. Worst fit

<table>
<thead>
<tr>
<th>A. First fit</th>
<th>B. Best fit</th>
<th>C. Worst fit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating System</td>
<td>Operating System</td>
<td>Operating System</td>
</tr>
<tr>
<td>Process 1</td>
<td>Process 1</td>
<td>Process 1</td>
</tr>
<tr>
<td>New process</td>
<td>Empty 60 blocks</td>
<td>Empty 60 blocks</td>
</tr>
<tr>
<td>Process 2</td>
<td>Process 2</td>
<td>Process 2</td>
</tr>
<tr>
<td>Process 3</td>
<td>Process 3</td>
<td>Process 3</td>
</tr>
<tr>
<td>Empty 52 blocks</td>
<td>New process</td>
<td>Empty 52 blocks</td>
</tr>
<tr>
<td>Empty 100 blocks</td>
<td>Empty 100 blocks</td>
<td>Empty 48 blocks</td>
</tr>
</tbody>
</table>
57. If a logical address in a paged memory management system is \( <2, 133> \), what do the values mean?
   This address means the 133 byte on page 2.

Exercises 58 - 60 refer to the following PMT.

<table>
<thead>
<tr>
<th>Page</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frame</td>
<td>5</td>
<td>2</td>
<td>7</td>
<td>3</td>
</tr>
</tbody>
</table>

58. If the frame size of 1024, what is the physical address associated with the logical address \( <2, 85> \)? 7253
59. If the frame size of 1024, what is the physical address associated with the logical address \( <3,555> \)? 3627
60. If the frame size of 1024, what is the physical address associated with the logical address \( <3,1555> \)? Illegal address. The offset is larger than the page size.
61. What is virtual memory and how does it apply to demand paging?
   Virtual memory is the illusion that memory is limitless and thus there is no limit on the size of a program. Demand paging is the technique where pages are brought into memory only when they are referenced (needed). Demand paging allows programs of any size, thus giving the illusion of infinite memory.
62. What are the conceptual stages through which a process moves while being managed by the operating system?
   new, ready, running, waiting, and terminated
63. Describe how a process might move through the various process states. Create specific reasons why this process moves from one state to another.
   A new process begins in the new state. When the process has no bars to its execution, it moves into the ready state. It waits in the ready state until it gets time in the running state. It runs for a while and issues a command for file input. The process is moved into the waiting state until the I/O has been completed, at which time it moves into the ready state to await another turn in the running state. Eventually it gets back to the CPU and runs until it needs access to a part of the program that is on secondary storage. It moves into the waiting state until the needed pages are brought in; then it moves back to the ready state. It gets its third shot at the CPU and finishes, and moves into the terminated state.
64. What is a process control block?
   A process control block (PCB) is a data structure that contains information about a process. A PCB is created for each new process. When a process moves from one state to another, its PCB is moved with it.
65. How is each conceptual stage represented in the OS?
   Each conceptual stage is represented by a list of the PCBs in that stage.
66. What is a context switch?
   When a process is moved out of the CPU, the current contents of the registers including the program counter must be saved in the process’s PCB. When a new
process moves into the CPU, the contents of the registers from this process's PCB are restored. This process of saving and restoring registers is called a context switch.

67. Distinguish between preemptive scheduling and nonpreemptive scheduling.

With nonpreemptive scheduling, once a process is in the running state it remains there until it voluntarily leaves. With preemptive scheduling, the OS can move a process from the running state to the waiting state or ready state.

68. Name and describe three CPU scheduling algorithms.

**First-come, first-served:**
The processes are moved into the running state in the order in which they arrive in the ready.

**Shortest job next:**
When the CPU is ready for another job, the process in the ready state that takes the shortest time is moved into the running state. The estimated length of time that a process needs the CPU may or may not be accurate.

**Round robin:**
Each process stays in the running state for a predetermined amount of time, called a time slice. When a process's time slice is over, it is moved back into the ready state, where it stays until it is its turn again for the CPU.

Use the following table of processes and service time for Exercises 69 through 71.

<table>
<thead>
<tr>
<th>Process</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service time</td>
<td>120</td>
<td>60</td>
<td>180</td>
<td>50</td>
<td>300</td>
</tr>
</tbody>
</table>

69. Draw a Gantt chart that shows the completion times for each process using first-come, first-served CPU scheduling.

0 120 180 360 410 710

P1 P2 P3 P4 P5

70. Draw a Gantt chart that shows the completion times for each process using shortest-job-next CPU scheduling.

0 50 110 230 410 710

P4 P2 P1 P3 P5

71. Draw a Gantt chart that shows the completion times for each process using round-robin CPU scheduling with a time slice of 60.

72. Distinguish between fixed partitions and dynamic partitions.
The sizes of the partitions are fixed in a fixed partition scheme, although they are not necessarily the same size. In a dynamic partition scheme, the partitions are allocated as needed.
**Question 1**
Consider a pure paging system that uses three levels of page tables and 64-bit addresses. Each virtual address is the ordered set \( v = (p, m, t, d) \), where the ordered triple \( (p, m, t) \) is the page number and \( d \) is the displacement into the page. Each page table entry is 64 bits (8 bytes). The number of bits that store \( p \) is \( n_p \), the number of bits that store \( m \) is \( n_m \) and the number of bits to store \( t \) is \( n_t \).

If \( n_p = n_m = n_t = 18 \)

A page number is 18+18+18= 54 bits long. Displacement \( d \) is 64-54=10 bits or 1K (1024).

1. How large is the table at each level of the multilevel page table?

The table at each of 3 levels is 18 bits long. This means a table points to \( 2^{18} = 256 \) K pages. Thus, the minimum table size is \( 256\times 2^{10}\times 8 \) (for 64 bit addressing) = 2048 Kbytes = 2 Mbytes

2. What is the page size, in bytes?

Since \( d \) is 10 (1K), the page size is 1 Kbytes (1 byte per location of page)

If \( n_p = n_m= n_t =14 \)

A page number is 14+14+14 = 42 bits long. Displacement \( d \) is 64-42=22 bits long or 4 M.

3. How large the table at each level of the multilevel page table?

Each level is 14 bits long. So a table points to \( 2^{14} = 16 \) K pages. Minimum table size is \( 16\times 2^{10}\times 8 = 128 \) Kbytes

4. What is the page size, in bytes?

\( d \) is 22 bits, the page size is 4Mbytes

**Question 2**
A system receives a series of page references in the following order: 1, 1, 3, 5, 2, 2, 6, 8, 7, 6, 2, 1, 5, 5, 5, 1, 4, 9, 7, 7. The system has five page frames. If all of the frames are initially empty, calculate the number of page faults using each of these algorithms:

a. **FIFO**

| 1 | 1 | 1 | 3 | 5 | 2 | 2 | 6 | 8 | 7 | 6 | 2 | 1 | 5 | 5 | 5 | 1 | 4 | 9 | 7 | 7 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 9 | 9 | 9 |
| 3 | 3 | 3 | 3 | 3 | 3 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 |
| 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
b. **LRU (Replace the page which has been unused for the longest time)**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
</tbody>
</table>

**Question 3**

A computer whose processes have 1024 pages in their address spaces keeps its page tables in memory. The overhead required for reading a word from the page table is 5 nsec. To reduce this overhead, the computer has a TLB, which holds 32 (virtual page, physical page frame) pairs, and can do a look up in 1 nsec. What hit rate is needed to reduce the mean overhead to 2 nsec?

**Solution:**

The effective instruction time is \(1h + 5(1 – h)\), where \(h\) is the hit rate. If we equate this formula with 2 and solve for \(h\), we find that \(h\) must be at least 0.75.

**Explain the difference between internal and external fragmentation?**

Internal Fragmentation is the area in a region or a page that is not used by the job occupying that region or page. This space is unavailable for use by the system until that job is finished and the page or region is released. The main difference is the allocation: internal fragmentation is allocated area and unused but external fragmentation is unallocated area and unused.

**Given five memory partitions of 100 KB, 500 KB, 200 KB, 300 KB, and 600 KB (in order), how would each of the first-fit, best-fit, and worst-fit algorithms place processes of 212 KB, 417 KB, 112 KB, and 426 KB (in order)? Which algorithm makes the most efficient use of memory?**

Let p1, p2, p3 & p4 are the names of the processes

a. **First-fit:**
   - P1>> 100, 500, 200, 300, 600
   - P2>> 100, 288, 200, 300, 600
   - P3>> 100, 288, 200, 300, 183
     - 100, 116, 200, 300, 183 <<<<< final set of hole
   - P4 (426K) must wait

b. **Best-fit:**
   - P1>> 100, 500, 200, 300, 600
   - P2>> 100, 500, 200, 88, 600
   - P3>> 100, 83, 200, 88, 600
   - P4>> 100, 83, 88, 88, 600
     - 100, 83, 88, 88, 174 <<<<< final set of hole
c. Worst-fit:
   - \( P_1 \gg 100, 500, 200, 300, 600 \)
   - \( P_2 \gg 100, 500, 200, 300, 388 \)
   - \( P_3 \gg 100, 83, 200, 300, 388 \)
   - \( 100, 83, 200, 300, 276 \) <<<<< final set of hole
   - \( P_4 \) (426K) must wait

In this example, Best-fit turns out to be the best because there is no wait processes.

Consider a logical address space of 64 pages of 1024 words each, mapped onto a physical memory of 32 frames.

a. How many bits are there in the logical address?
b. How many bits are there in the physical address?

Method1:
   a) \( m = \) ???
      Size of logical address space = \( 2^m = \) # of pages \( \times \) page size
      \( 2^m = 64 \times 1024 \)
      \( 2^m = 2^6 \times 2^{10} \)
      \( 2^m = 2^{16} \) »» \( m=16 \) bit

Method2:
   \( m = \) ???
   # of pages = \( 2^{m-n} \)
   \( n = \) ???
   Page size=\( 2^n \)
   \( 1024=2^n \)
   \( 2^{10}=2^n \) »» \( n=10 \) bit
   Again: # of pages = \( 2^{m-n} \)
      \( 64=2^{m-10} \)
      \( 2^6=2^{m-10} \)
      \( 6= m-10 \) »» \( m=16 \) bit

b)
   Let \( (x) \) is number of bits in the physical address
   \( x = \) ???
   Size of physical address space = \( 2^x \)
   Size of physical address space = # frames \( \times \) frame size
      (frame size = page size)
   Size of physical address space = \( 32 \times 1024 \)
      \( 2^x = 2^5 \times 2^{10} \)
      \( 2^x = 2^{15} \)
   »» number of required bits in the physical address=\( x =15 \) bit

Consider a logical address space of 32 pages of 1024 words per page, mapped onto a physical memory of 16 frames.
   a. How many bits are required in the logical address?
   b. How many bits are required in the physical address?
a) \( m = \)???
Size of logical address space = \( 2^m = \) # of pages \( \times \) page size
\[ = 32 \times 1024 = 2^{15} \implies m=15 \text{ bit} \]

b) Size of physical address space = # of frames \( \times \) frame size
(frame size = page size )

Size of physical address space = \( 16 \times 1024 = 2^{14} \)
\( \implies \) number of required bits in the physical address =14 bit

Assuming a 1-KB page size , What are the page numbers and offsets for the following address references (provided as decimal numbers)

\begin{align*}
a. & \quad 2375 & d. & \quad 256 \\
b. & \quad 19366 & e. & \quad 16385 \\
c. & \quad 30000
\end{align*}

Answer:
Page size =2\(^n\) =1024 B = 2\(^{10}\) B
# of bits in offset part (n) =10

Solution steps :
1. Convert logical address: Decimal \( \rightarrow \) Binary
2. Split binary address to 2 parts (page #, Offset), offset : n digits
3. Convert offset & page# : Binary \( \rightarrow \) Decimal

<table>
<thead>
<tr>
<th>Logical address (decimal)</th>
<th>Logical address (binary)</th>
<th>Page # (6 bits) (binary)</th>
<th>Offset (10 bits) (binary)</th>
<th>Page # (decimal)</th>
<th>Offset (decimal)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2375</td>
<td>0000 1001 0100 0111</td>
<td>0000 10</td>
<td>01 0111</td>
<td>2</td>
<td>327</td>
</tr>
<tr>
<td>19366</td>
<td>0100 1011 1010 0110</td>
<td>0100 10</td>
<td>11 0110</td>
<td>18</td>
<td>934</td>
</tr>
<tr>
<td>30000</td>
<td>0111 0101 0011 0000</td>
<td>0111 01</td>
<td>01 0000</td>
<td>29</td>
<td>304</td>
</tr>
<tr>
<td>256</td>
<td>0000 0001 0000 0000</td>
<td>0000 00</td>
<td>01 0000</td>
<td>0</td>
<td>256</td>
</tr>
<tr>
<td>16385</td>
<td>0100 0000 0000 0001</td>
<td>0100 00</td>
<td>00 0001</td>
<td>16</td>
<td>1</td>
</tr>
</tbody>
</table>

Consider a paging system with the page table stored in memory.

a. If a memory reference takes 200 nanoseconds, how long does a paged memory reference take?
b. If we add associative registers, and 75 percent of all page-table references are found in the associative registers, what is the effective memory reference time? (Assume that finding a page-table entry in the associative registers takes zero time, if the entry is there.)

Answer:
a. memory reference time= 200+200= 400 ns
   (200 ns to access the page table in RAM and 200 ns to access the word in memory)

b. **Case (1) : page entry found in associative registers (part1)**
   Memory access time = 0+200=200 ns
   (0 ns to access the page table in associative registers and 200 ns to access the word in memory)

   **Case (2) : page entry NOT found in associative registers (part1) but found in page table in RAM**
   Memory access time = 0+200+200=200 ns
   (0 ns to access the page table in associative registers (part1), 200 ns to access the page table(part2) in RAM and 200 ns to access the word in memory)

>>> Effective access time = \[\sum \text{[probability of the case} \times \text{access time of this case]}\]

**Effective access time = [0.75 \times 200 ]+ [0.25 \times 400]= 250 ns.**

Consider a computer system with a 32-bit logical address and 4-KB page size. The system supports up to 512MB of physical memory. How many entries are there in each of the following:

a. A conventional single-level page table

b. An inverted page table
   a) # of pages= # of entries = ????
   Size of logical address space = \(2^m = \# \text{ of pages} \times \text{page size}\)
      \[2^{32} = \# \text{ of pages} \times 2^{12}\]
      \# of pages = \(2^{32} / 2^{12} = 2^{20}\) pages

b) Size of physical address space = \# of frames \times frame size
   (frame size = page size ) \[\Rightarrow \Rightarrow \Rightarrow \] In paging in general
   \[\# \text{ of frames} = \# \text{ of pages}\] \[\Rightarrow \Rightarrow \Rightarrow \] In Inverted paging ONLY
   \[2^{29} = \# \text{ of frames} \times 2^{12}\]
   \# of frames = \(2^{29} / 2^{12} = 2^{17}\) pages

   Number of entries= = \(2^{17}\) pages
8.21 Consider the following segment table:

<table>
<thead>
<tr>
<th>Segment</th>
<th>Base</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>219</td>
<td>600</td>
</tr>
<tr>
<td>1</td>
<td>2300</td>
<td>14</td>
</tr>
<tr>
<td>2</td>
<td>90</td>
<td>100</td>
</tr>
<tr>
<td>3</td>
<td>1327</td>
<td>580</td>
</tr>
<tr>
<td>4</td>
<td>1952</td>
<td>96</td>
</tr>
</tbody>
</table>

What are the physical addresses for the following logical addresses?

a. 0,430  
b. 1,10  
c. 2,500  
d. 3,400  
e. 4,112

- a. 219 + 430 = 649
- b. 2300 + 10 = 2310
- c. illegal reference, trap to operating system
- d. 1327 + 400 = 1727
- e. illegal reference, trap to operating system

8.11 Compare paging with segmentation with respect to the amount of memory required by the address translation structures in order to convert virtual addresses to physical addresses.

**Answer:** Paging requires more memory overhead to maintain the translation structures. Segmentation requires just one (or more) entry for code and one entry for data. Paging on the other hand requires multiple entries for code and data depending on page size. So, paging almost requires more memory for page table than the memory space required to segment table.