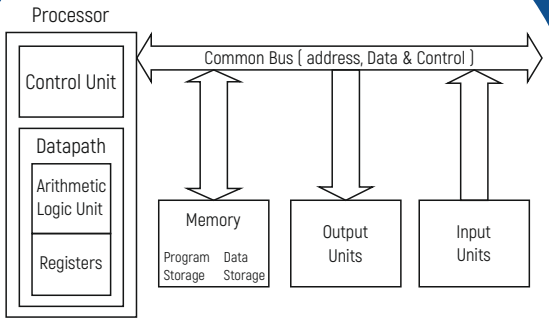
ACE

ESE | GATE | PSUs

ELECTRONICS & TELECOMMUNICATION ENGINEERING

COMPUTER ORGANIZATION & ARCHITECTURE

Study Material with Classroom Practice Questions



Chapter 1 Computer Arithmetic

(Solutions for Text Book Practice Questions)

Objective Practice Solutions

01. Ans: (a)

Sol: E = e + Bias

03. Ans: (a)

Sol: X = –14.25, S =1, e = original exponent

E = biased Exponent,

b = biasing amount

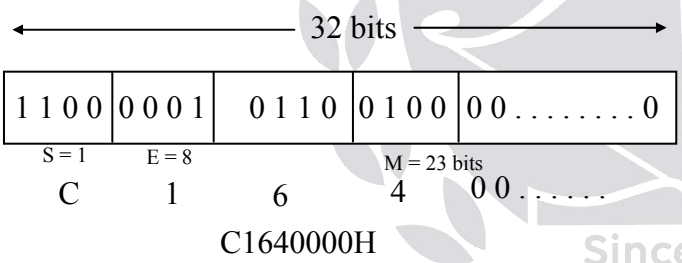
$14.25 = 1110.01 \times 2^0$

$= 1.11001 \times 2^3$

e = 3, E = 3 + 127

= 130

130 = 10000010 M = 1100100…0 (23 bits)



C 1 6 4 0 0 . . . . . .

C1640000H

Conventional Practice Solutions

01.

Sol:

S	E	M	value
1	8	23	
0/1	000000000	000...0	± 0
	E = 0	M = 0	
0/1	11111111	000....0	± ∞
	E = 255	M = 0	

S	E	M	Value
0/1	E ≠ 0 & E ≠ 255 1≤ E ≤ 254	xxx	Implicit number
0/1	E = 0	M ≠0	Fractional number
0/1	E = 255	M≠0	NAN (Not A Number)

Chapter 2 Memory Organization & Secondary Memory

Objective Practice Solutions

01. Ans: (c)

Sol: T = 200 ns

$\therefore$ word frequency

$$= \frac{1}{200\times10^{-9}} = 5\times10^{6}\,\text{words/sec}$$

02. Ans:

Sol: (a) Capacity = T × S ×B

$= 512 \times 2048 \times 4096 \times B$

$= 2^{9} \times 2^{11} \times 2^{12}\,B$

$= 2^{32}\,B = 4\,GB$

(b) Data Transfer rate

1- Revolution disk covers = $2^{11}.\,2^{12}$

= 8 MB

Disk speed 3000 rpm

= 3000 revolutions in one minute

One minute = 60 sec = 60 ×1000ms

= 60,000 ms

3000 revolutions → 60,000 ms

1 revolution →?

$$= \frac{60000\text{m}}{3000} = 20\text{ ms}$$

20 ms → 8 MB

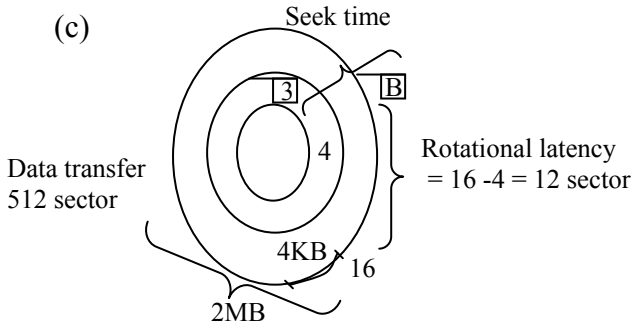
1000 ms→ ?

$$= \frac{8000\text{ M}\times\text{m}}{20\,\text{m}} = 400\,\text{MB}$$

Data Transfer rate = 400 MB /Sec

= 400 MBPS

(c)



Data transfer 512 sectors

$$\frac{2\text{MB}}{4\text{KB}} = \frac{2^{21}}{2^{12}} = 2^{9}$$

$T_{file} = T_{seek\ time} + T_{rotational\ latency}$ + Data Transfer time

(512 + 12) sectors are to be covered

20 ms → 2048 sectors

? → 524 sectors

Sectors requires = 5.117 ms

T_{file} = 10 ms + 5.117 ms = 15.117 ms

03. Ans: (a)

Sol: ROM is used for function table with large size because after program; ROM content is not possible to destroy and design cost is cheap.

04. Ans: (d)

Sol: **Cache memory** - A special very high speed memory called a cache is sometimes used to increase the speed of processing by making current programs and data available to the CPU at a rapid rate. The cache memory is employed in computer systems to compensate for the speed deferential between main memory access time and processor Logic.

05. Ans: (c)

Sol: The use of a cache in a computer system increases the Average speed of memory access.

08. Ans: (b)

Sol: Average Latency time= ½ rotation time

Speed = 7200 rpm ⇒ 120 rps

For 1 rotation = 1/120 = 8.33ms

For ½ rotation = 4.166 ms

09. Ans: (a)

Sol: **LIFO:** Last In First Out

10. Ans: (d)

Sol: Hit ratio is defined as the number of hits divided by the total number of CPU references to the memory (hits plus misses)

$T_{avg} = H\,C + (1 - H)M$

Where H = hit ratio of cache memory

C = time to access information in cache memory

M = Miss penalty

= main memory access time + cache memory access time.

11. Ans: (c)

Sol: DRAM access much slower than SRAM

- More bits → longer wires
- Buffered access with two level addressing
- SRAM access latency : 2 - 3 ns
- DRAM access latency: 20 - 35 ns
- Static RAM -10 ns
- Hard disk for personal computers boast access times of about a to 15ms
- 200 times slower than average DRAM.

12. Ans: (d)

Sol: Cache size = 4 K word

One set has 4 blocks

$\therefore$ Number of sets $= \dfrac{\text{Total number of blocks}}{4}$

Number of words $= 2^6$

$\Rightarrow W = 6$

$$\text{Number of cache blocks} = \frac{4K}{64} = \frac{2^{12}}{2^6} = 2^6$$

$$\text{Number of cache sets} = \frac{2^6}{4} = 16 = 2^4$$

$S = 4$

13. Ans: (a)

Sol: Access time = Seek time + Average rotation delay

1 rotation time = 33.3 ms

$\therefore$ Average rotation time = 16.7 ms

Total access time = 46.7 ≈ 47 ms

14. Ans: (c)

Sol: Cache memory is used to improve the overall system performance.

15. Ans: (a)

Sol: Word size = 16-bit

Access time = 80-ns

No of bytes to be transferred = 1024-B

= 512 Words.

Total time = 512 × 80ns 96μs

= 40.96 μs

16. Ans: (c)

Sol: $2^{12} \times 8 = 4096 \times 8$

$= 32768.$

17. Ans: (c)

Sol: Remain same because it is non-volatile memory.

18. Ans: (d)

Sol: Hit Rate = 80%

$T_{CM} = 10ns$, $T_{MM} = 100ns$

$T_{Avg} = H \times T_{CM} + (1-H) \times T_{MM}$

$= (0.8 \times 10ns) + 0.2 \times 100ns$

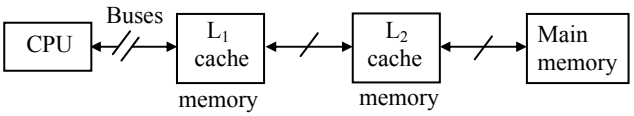
$= 8ns + 20ns$

$= 28$ ns

Conventional Practice Solutions

01.

Sol:



- Always L_1 is the fastest memory and L_2 is slower memory
- While executing a program, the CPU tries to read the instruction from L_1 cache first; if the searched word is available then operation is Hit, otherwise operation is miss.
- If there is Hit in 2^{nd} level memory then a block is transferred from L_2 cache to L_1 cache and then a word is transferred from L_1 cache to CPU.
- If again there is a miss in L_2 cache memory; then immediately a block is transferred from main memory to L_2 cache and then L_2 cache to L_1 cache

Performance of the cache memory is measured in terms of Hit rate

$$\text{Hit Rate} = \frac{\text{Number of Hits given in the cache}}{\text{Total number of memory visits by CPU}}$$

- More Hit Rate gives more performance
- When the searched word is available in searched memory, then operation is Hit, otherwise miss.

Chapter 3 Pipeline Organization

Objective Practice Solutions

02. Ans: (b)

Sol: First task will be completed after n clocks (because there are n segments) and the remaining m-1 tasks are shipped out at the rate of one task per pipeline clock.

Therefore, n + (n–1) clock periods are required to complete m tasks using an n-segment pipeline. If all m tasks are executed without any overlap, mn clock periods are needed because each task has to pass through all n segments. Thus speed gained by an n segment pipeline can be shown as follows:

Speed up P(n) =

$$\frac{\text{number of clocks required when there is no overlap}}{\text{number of clocks required when tasks are overlapped in time}}$$

$$= \frac{mn}{n+m-1}$$

03. Ans: (c)

Sol: Max. stage delay = 160 μs

Buffer delay = 5 μs

Pipeline clock = 165 μs

$T_{1000} = (K + n - 1)\, T_p$ clock

$$= (4+999) * 165 = \left(\frac{165495}{1000}\right) \mu s$$

$$= 165.5\ \mu s$$

04. Ans: (b)

Sol: For D_1 processor, maximum $T_{seg} = 4$ ns,

n = 100, k = 5

Time = 104 × 4 ns = 416 ns

For D_2 processor,

n = 100, k = 8, $T_{seg} = 2$ ns

Time = 107 × 2 ns = 214 ns

Hence, 202 ns time will be saved

Conventional Practice Solutions

01.

Sol: Speed Up:

The Speed Up of a pipeline processing over an equivalent non pipeline processing is defined by the ratio

$$S = \frac{\text{time without pipeline}}{\text{time with pipeline}} = \frac{n\, t_n}{(K+n-1)t_p}$$

Where 't_n' is the time taken to complete an instruction without pipeline.

't_p' is the time taken for a stage, which is maximum of all stages.

Speed Up factor can also given by

$$S = \frac{\text{Pipeline depth}}{1 + \text{Pipeline stall cycles/ instruction}}$$

For ideal case,

Pipeline stall cycles / instruction is **zero**.

S = Pipeline depth = number of stages

Chapter 4 I/O Organization

Objective Practice Solutions

01. Ans: (b)

Sol: External interrupts comes from input-output devices, from a timing device, from a circuit monitoring the power supply or internal from other external source.

Example: I/O devices requesting transfer of data

- A DMA controller manages the data transfer
- A DMA controller can be implemented as separated controller from the processor or integrated controller in the processor.
- Based on the request length, the DMA controller optimizes transfer performance between source and destination with different external data by widths

02. Ans: (d)

Sol: The DMA is the most suitable for Hard disk.

03. Ans: (b)

Sol: On receiving an interrupt from an I/O device, the CPU branches off to the interrupt service routine after completion of the current instruction.

04. Ans: (c)

Sol: In microprocessor based systems DMA facility is required to increase the speed of data transfer between memory and the I/O devices.

05. Ans: (b)

Sol: An I/O processor controls the flow of information between main memory and I/O devices. It is the extension of concept of DMA.

06. Ans: (b)

Sol: For vectored hardware interrupt, the interrupting device supplies the respective address with additional hardware.

07. Ans: (a)

Sol: In a microprocessor based system, isolated I/O method is similar to "I/O mapped I/O" where the memory and I/O devices are provided with separate address space and address decoders.

08. Ans: (b)

Sol: Internal interrupts in a microprocessor based system are initiated by error conditions (Ex. Divide by zero overflow conditions). External interrupts are produced by external hardware which may occur at any point of time.

Conventional Practice Solutions

01.

Sol: Cycle Stealing Mode:

An alternative technique called cycle stealing allows the DMA controller to transfer one data word at a time, after which it must return control of the buses to the CPU. The CPU merely delays its operation for one memory cycle to allow the direct memory I/O transfer to "steal" one memory cycle.

Chapter 5 Instruction Set & Addressing Modes

Objective Practice Solutions

01. Ans: (a)

Sol: 1. Direct y = z[I]

2. Immediate x = 5

3. Index x = z + m

4. Auto k = c+ +

02. Ans: (c)

Sol: Register indirect addressing lets you generate lots of different addresses when a program is executed. Address register indirect addressing is so called because it uses an address register to point at the location of the operand in memory; that is, the address of an operand is obtained indirectly via an address register. Instead of telling the computer where the operand is, you tell it where the address of the operand can be found.

03. Ans: (c)

Sol: **Relative Addressing mode**: Relative Addressing mode is used in intra segment branching.

Effective Address = Program Counter + Displacement

Relative Base Addressing Mode: Effective Address = PC + BR + displacement . Where PC stands for program counter and BR stands for base register.

"The code is dependent means the option is direct addressing mode".

04. Ans: (b)

Sol: Base register addressing mode is used to relocate the program from one segment to other segment.

Conventional Practice Solutions

01.

Sol: (a) MAR: It is a memory address register that holds the Address of the memory Register

- In Basic processor, it is connected to the address Bus and it's size is equal to the address Bus size
- μp places the memory address in this register while performing memory read and memory write operation.
- During instruction fetch; MAR is used to hold program address and during data read or Data write operations MAR holds the Data memory Register Address.

(b) MDR: It is used to hold the content of the memory Register while performing Read/write operation from memory

- It is connected to Data Bus.
- It's size is equal to the Data bus size while fetching the instruction, initially opcode of the instruction is placed in MDR

(c) PC: It is program counter that holds the address of the next instruction to be fetched; It's size is equal to the address bus size of the processor. After fetching an instruction, PC content is automatically incremented to point the address of the next instruction to be fetched.

(d) IR: It is an instruction Register that holds the opcode of the instruction after it's fetching

After fetching an instruction, opcode will be placed in IR (from MDR). later it sends to control Register for completing it's Decode and execution.

Size of the IR equal to the MDR size

Chapter 6 C Language

Objective Practice Solutions

01. Ans: (a)

Sol: A program that translates a high-level language program into a object module is called a complier.

02. Ans: (b)

Sol: double is used to define BIG floating point numbers. It reserves twice the storage for the number.

03. Ans: (a)

Sol: Characters are assigned variables by using single quotes, where strings we use double quotes.

04. Ans: (c)

Sol: #define SALES_TAX_RATE 0.0825 is a valid constant

05. Ans: (c)

Sol: printf ("%d%c%f", 23, 'z', 4.1);

07. Ans: (b)

Sol: The value of x is decremented by one after the print statement.

08. Ans: (c)

Sol: Left side of assignment operator allows only one variable, not a expression

In option (b) the meaning of

a * = b gives a = a * b

In option (d) b = 0 is first evaluate the result is assign to 'a' because '=' operator is right to left associativity.

Finally option (c) is invalid

09. Ans: (b)

Sol: Except option (b) remaining all expressions having different precedence levels but in option (b) '*', '/' operators are having same precedence levels, so for evaluation we need associativity. Here associativity is 'left to right'.

10. Ans: (a)

Sol: Option (b), (c), (d) are logical operators. Where as option (a) is a statement.

11. Ans: (a)

Sol: Option (b), (c), (d) are relational operators which compare the two variables and gives result in the form of boolean data type i.e. true (or) false.

Where as option (a) is assignment operators which assign the value to variable.

12. Ans: (d)

Sol: The complement of equal (= =) operator is not equal (! =)

13. Ans: (b)

Sol: for, while and do……while are looping constructs but switch case is a statement.

14. Ans: (c)

Sol: Before entering into the loop while checks the condition i.e., while is pre-test loop, where 'do-while' is post test loop.

15. Ans: (a)

Sol: The address of (&) operator is used to extract the address for a variable.

16. Ans: (c)

Sol: The syntax int *p; is used to declare a pointer variable to an integer.

17. Ans: (c)

Sol: For creating a pointer we use '*' and address of variable we use '&'.

18. Ans: (c)

Sol: We can increment the pointer value and comparing but we can't divide prt + 5 is the pointes value 5 elements away. ++ ptr pre-increment of pointer & that points to next address.

19. Ans: (b)

Sol: A variable length string can be controlled by a length or a delimiter. The C language uses a null to terminate variable length strings.

20. Ans: (c)

Sol: The statement **employee.name** is used to declare the name of the employee.

21. Ans: (b)

Sol: for loop iterate 10 times because every time 'i' value getting to double, means 'i' initially 1, then 2, 4, 8, 16, 32, 64, 128, 256, 512 finally 1024 give false value to for loop. So total 10 times of iteration.

22. Ans: (c)

Sol: For every 'i' value 'j' loop iterates 10 times and 'i' will iterate 10 times, So $10 \times 10 = 100$ iterations possible.

23. Ans: (b)

Sol: Both statements are the definitions of break and continue respectively.

24. Ans: (b)

Sol: **Loop:** In computer programming, a loop is a sequence of instructions that is continually repeated until a certain condition is reached.

Conventional Practice Solutions

01.

Sol: Program to find biggest of 'n' integers. (Input the numbers continuously until the user requests to stop.)

```
#include<stdio.h>
void main()
{
    int n, b = 0;
    do /* start of do-while loop*/
    {
            printf("\n enter a number:");
            scanf("%d", &n);
            if(b ==0)
                    /* if this is 1st number */
            b=n;
                    /* consider it as biggest */
            else
               /* from 2nd number onwards */
            if(n>b)
/* if the number is bigger than biggest*/
            b=n;
          /*then only consider it as biggest */
printf("\n Do you want to enter another number (Y/N)?");
     }      while (getchar () = = 'Y');
   /* get choice & repeat if YES */
    printf("The biggest number is %d\n", b);
}/* end of main */
```

In the above program, the loop is used to accept numbers continuously until the user says NO (choice other than 'Y'). We know that asking the choice for next number should be done only after taking a number. Therefore, in this situation the do-while loop is more suitable than the while loop.

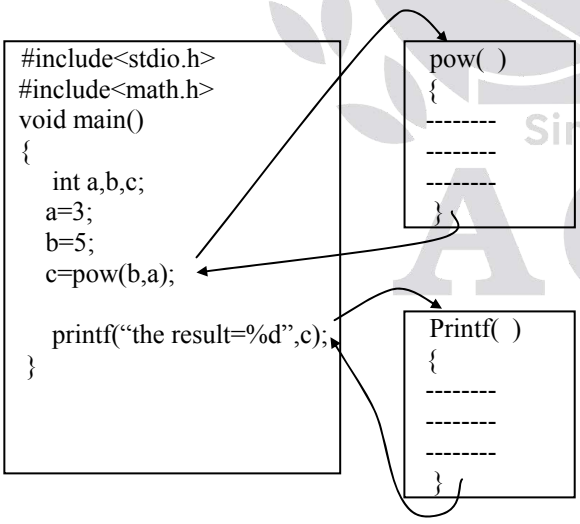
02.

Sol: Function is a sub-program as it also stored separately in the memory and executed wherever it is called. Here main( ) is treated as a boss function and other functions are subordinates to main( ). The main function is called by operating system, which is indirectly by the user.

Let us see the following example

```
#include<stdio.h>
#include<math.h>
void main ( )
    {   int a,b,c;
        a = 3;   b = 5;
        c = pow (b, a);
        printf ("the result = %d", c);
    }
```

The execution of functions (jumping & returning) is as follows. (this pow & printf are library functions they attached at compile time from C-software (*.lib) files.)



While jumping control to pow( ) function, the values b&a are passed as input, later returns calculated **power** value and assigned to **'c'**. Whereas, while jumping into printf( ) function, the string constant **"the result = %d"** & **'c'** are passed as input, later printf prints this input on the screen. It won't return any value, because its job is simply printing message on screen.

03.

Sol: Expressions

An expression is a systematic combination of operands and operators to specify certain calculations. However, any single constant, variable, or any valued single item is also called expression.

C language allows us to use complex expressions, where such expressions should be cared, otherwise it lead to logical errors. Let us see, how expression should be typed in language syntax.

For example:

1. $ax^3 + bx^2 + cx + d$

Ans: a * x * x * x + b * x * x + c * x + d

2. $\dfrac{-b + \sqrt{b^2 - 4ac}}{2a}$

Ans: (–b + sqrt (b*b–4*a*c)) / (2*c)

Here sqrt () is a function, finds square root of given value.

3. $\dfrac{ax^{45} + 6x^5}{cx + 7}$

Ans: (a*pow (x, 45) + 6*pow (x, 5)) / (c*x +5)

Chapter 7 Basics of Operating Systems

Objective Practice Solutions

01. Ans: (a)

Sol: Multi programming by definition is ability of OS to manage multiple ready to run program in memory. Remaining all other options are not correct/ Invalid.

02. Ans: (d)

Sol: Programmers can access OS resources through system call routines..

03. Ans: (d)

Sol: Presence of multiple CPU's is multiprocessor OS, where as multiprogramming can be possible with only one CPU.

04. Ans: (d)

Sol: Interrupts are used for Invoking system calls, Indicating complication of I/O, Sometimes pre-empting running process from CPU.

05. Ans: (c)

Sol: A processor in the context of computing a piece of hardware that executes a set of instructions.

06. Ans: (d)

Sol: The basic goal of multiprogramming is to keep the devices along with the CPU busy.

07. Ans: (b)

Sol: Zombie implies the process entry still exists in the process table maintained by OS.

08. Ans: (c)

Sol: Processes are generally swapped out from memory to Disk when they are decided to be suspended.

09. Ans: (b)

Sol: Suppose a Processor does not have any Stack Pointer Register It can have subroutine call instruction, but no nested subroutine calls.

10. Ans: (d)

Sol: When an Interrupt occurs, an Operating System May change state of interrupted process to 'blocked' and schedule another process.

Conventional Practice Solutions

01.

Sol: An Operating System (OS) is an interface between a computer user and computer hardware. An operating system is a software which performs all the basic tasks like file management, memory management, process management, handling input and output, and controlling peripheral devices such as disk drives and printers.

Some popular Operating Systems include Linux Operating System, Windows Operating System, VMS, OS/400, AIX, z/OS, etc.

Definition:

An operating system is a program that acts as an interface between the user and the computer hardware and controls the execution of all kinds of programs.

Following are some of important functions of an operating System.

- Memory Management
- Processor Management
- Device Management
- File Management
- Security
- Control over system performance
- Job accounting
- Error detecting aids
- Coordination between other software and users

02.

Sol: An operating system is the most important software that runs on a computer. It manages the computer's memory and processes, as well as all of its software and hardware. It also allows you to communicate with the computer without knowing how to speak the computer's language. Without an operating system, a computer is useless.

Your computer's operating system (OS) manages all of the software and hardware on the computer. Most of the time, there are several different computer programs running at the same time, and they all need to access your computer's central processing unit (CPU), memory, and storage. The operating system coordinates all of this to make sure each program gets what it needs.

Chapter 8 Process Characteristics & Applications

Objective Practice Solutions

01. Ans: (d)
Sol: Process gets blocked when it requires I/O or request any resource. Remaining are done by OS.

02. Ans: (d)
Sol: Block initiated by process and Ready by OS.

03. Ans: (a)
Sol: Whenever an interrupt occurs, while process is running on CPU, it gets preempted and goes to ready once.

04. Ans: (a)
Sol: Process running in user mode has to increase the PC Contents.

05. Ans: (b)
Sol: Round Robin scheduling algorithm is especially used for time sharing system. It is similar to FCFS but preemption is added to switch between processes small unit of time called time Quantum or time slice is defined.

06. Ans: (d)
Sol: A critical region (section) is the portion of program text that involves shared variables.

07. Ans: (a)
Sol: Time of submitting a request to the time at which initial response is generated.

08. Ans: (a)
Sol: If the system is non-preemptive, then if a long process is running it will cause starvation to longer process.

09. Ans: (c)
Sol: Round Robin is non-priority based and does not need any knowledge of service times of processes.

10. Ans: (d)
Sol: Generally if multithreading is done at user level then the whole process gets blocked otherwise only the respective thread gets blocked.

11. Ans: (a)
Sol: CPU efficiency is a measure of useful computation over total computational work. Total work includes the time for context switching.

12. Ans: (b)
Sol: Round Robin being preemptive, with time quantum can result in reducing prolonged waiting of processes and thereby improves inter-activeness.

13. Ans: (b)
Sol: Real time environments are constrained by strict Deadlines. Hence priority based scheduling is required to challenge the deadlines.

14. Ans: (d)
Sol: Round Robin Scheduling is based on Time Quantum and arrival times of process.

15. Ans: (c)
Sol: Cycle stealing is used for DMA operation

16. Ans: (a)
Sol: Race condition implies that multiple processes are contending competitively or cooperatively to access critical resources.

17. Ans: (c)
Sol: Semaphores always satisfy all the conditions of synchronization requirements.

18. Ans: (c)

Sol: Processes are generally swapped out from memory to Disk when they are decided to be suspended.

19. Ans: (c)

Sol: RR is a pre-emptive scheduler, which is designed especially for time-sharing systems. In other words, it does not wait for a process to finish or give up control. In RR, each process is given a time slot to run. If the process does not finish, it will "get back in line" and receive another time slot until it has completed.

20. Ans: (b)

Sol: Total number of jobs executed per unit time (also called throughput)is maximum in SJF. Since shorter jobs are executed first and more number of jobs will be executed per unit time.

21. Ans: (b)

Sol: FCFS/FIFO is non-preemptive as preemption takes place only after completion of the process. Remaining algorithms are preemptive.

22. Ans: (b)

Sol: Once every process got scheduled one time each (for T time units).

All processes have now the same waiting time = (n–1) T (n = number of processes).

Till now, the scheduling was Round Robin with pre-emption at periods of T. The same analysis can be extended for further scheduling as all the processes have the same priority = (n – 1)T.

23. Ans: (c)

Sol: A CPU generally handles an interrupt by executing an interrupt service routine by checking the interrupt register after finishing the execution of the current instruction.

26. Ans: (c)

Sol: If the time-slice used in the round-robin scheduling policy is more than the maximum time required to execute any process, then the policy will Degenerate to first come first serve.

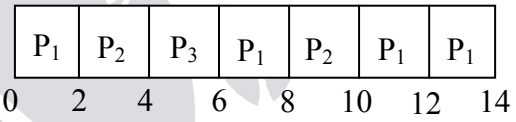
27. Ans: (a)

Sol: As given process scenario

Process_id	Arrival time	Burst time	Complete time
P_1	0	8	14
P_2	0	4	10
P_3	0	2	6

Preparing Gantt chart for Round-Robin scheduling, Q = 2

Complete time of processes P_1, P_2 and P_3 as 14, 10, 6.



Turnaround time of a process

= Complete time – Arrival time.

P_id	Turnaround time
P_1	14
P_2	10
P_3	6

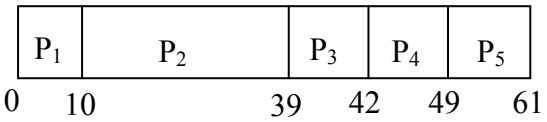
$$\text{Average turnaround time} = \frac{14+10+6}{3}$$
$$= 10\,\text{ms}.$$

28. Ans: (a)

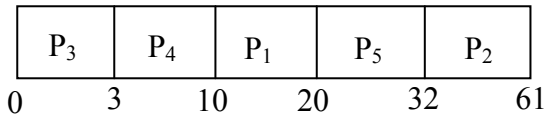
Sol: Given scenario of the processes

Process	Burst time
P_1	10
P_2	29
P_3	3
P_4	7
P_5	12

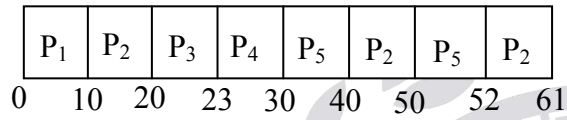
FCFS:



Non-Preemptive SJF



Round-Robin:



Average waiting times for FCFS, SJF and RR are 28 ms, 13 ms and 23 ms respectively.

Conventional Practice Solutions

01.

Sol: (a) Ready → Run ⇒ Transition is possible, when a process is scheduled and displaces to run in CPU.

(b) Ready → Blocked ⇒ Not possible A process can go into blocked state from running state only.

(c) Blocked → Ready ⇒ Transition is possible when a process completes the I/O or event, then it makes this transition.

(d) Running → Terminate ⇒ Transition is possible. When the execution of process is finished.

(e) Ready → Terminate ⇒ Transition not possible. A process can go to terminated state only from running state.

02.

Sol: The process will run twice as of many times it should run.

03.

Sol: **Deadlock:** Two or more processes wait for such an event which is never going to occur. Deadlocked processes can never execute further but a process in starvation may further after indefinite time-waiting.

Chapter 9 Memory Management

Objective Practice Solutions

01. Ans: (b)

Sol: Number of Memory Chips required = Memory Required/Chip Size; Each Row having 2 chips of dimension 256×4 will get memory of capacity 256 B; 128 rows each having 2 chips will organize memory of capacity 32 KB.

02. Ans: (d)

Sol: In board memory is like RAM and out board memory is like Hard-Disk and off-time storage is like magnetic tape.

03. Ans: (a)

Sol: The number of allocation units

$= 1GB/64\ KB = 2^{14}$.

1 bit is required for each allocation unit.

04. Ans: (a)

Sol: The size of the page $= 2^{32-(9+11)}$

$= 2^{12}$bytes = 4 KB

The number of pages

= Top level pages + second level pages

$= 2^{20}/2^{12}$ (Top level pages)

$+ 2^{20}$(second level pages)

$\approx 2^{20} = 1M$

05. Ans: (d)

Sol: Using Best fit the request for 120 K gets placed in 150 K and 10 K in 30 K and 250 K gets placed in 300 K; at this juncture 60 K can't get placed and hence the total left over free memory is 120 K.

08. Ans: (c)

Sol: L.A.S = 4K × 512 = 2^{21} $\Rightarrow$ L.A = 21 bits.

P.A.S = 1K×512 = 2^{19} $\Rightarrow$ P.A = 19 bits.

09. Ans: (b)

Sol: Use the equation:

$E_{mat} = x(c+m) + (1-x)(c+2m)$

10. Ans: (c)

Sol: Larger pages leads to less number of pages and less number of entries in page table.

11. Ans: (a)

Sol: The formula for optimal page size is $\sqrt{2.S.e}$ values of S = 32 MB; e = 4B

13. Ans: (c)

Sol: Larger Pages implies less number of pages.

14. Ans: (a)

Sol: Smaller Pages will involve less internal fragmentation.

17. Ans: (a)

Sol: Fixed partitioning technique limits or restrict the degree of multiprogramming to the number of partitions.

18. Ans: (a)

Sol: Using the equation

$E_{mat} = (1–P)\ m + P*S$;

Here m = 5 μs; S = 50 ms;

1–P = 0.9999 & P = 0.0001

Substituting in the above equation

E_{mat} = 10 μs

19. Ans: (b)

Sol: Due to Belady's anomaly, sometimes page fault increases with the increase in page fault rate.

20. Ans: (b)

Sol: The Number of page faults using optimal replacement is 11, giving a page fault rate of 50%, using the equation.

$E_{mat} = P*s + (1–p)*m$;

Where P = 50%; m = 1 ms; s = 30 ms;

21. Ans: (a)

Sol: Apply optimal replacement

22. Ans: (d)

Sol: All will not follow locality of reference.

23. Ans: (b)

Sol: High page fault rate implies thrashing which can be controlled by decreasing degree of multiprogramming and addition of more main memory.

24. Ans: (a)

Sol: It's the work of addressing modes to provide minimum required pages, OS has the work to provide more pages.

25.

Sol: Logical Address = 13 bits

Physical Address = 15 bits

26. Ans: (c)

Sol: Threads refer to the execution of different processes simultaneously in the CPU.

Virtual address space is implemented in Memory.

The File system in the way in which data in a disk is organized.

A signal is acted upon using the mechanism of interrupts.

29. Ans: (a)

Sol: Both statements are true and statement II is the correct explanation for statement I.

Conventional Practice Solutions

01.

Sol: EMAT = p*s + (1–p)*m

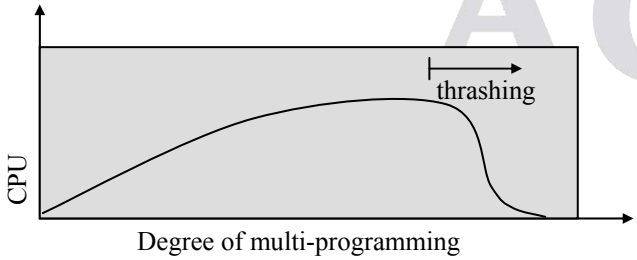
p is page fault rate, s is page fault service time, m is memory access time. In the question s = 5ms ; m = 120 ns; EMAT = 1 micro sec. p = 0.0176 %

02.

Sol: Thrashing:

It's a phenomenon where a process is busy swapping pages in and out. **OR** high paging activity is observed.

- Exhibits swapping out a piece of a process just before that piece is needed.
- The processor spends most of its time swapping pieces rather than executing user Instructions.
- How thrashing occurs:
  If a process does not have "enough" pages, the page-fault rate is very high. This leads to low CPU utilization; operating system thinks that it needs to increase the degree of Multiprogramming another process added to the system (as depicted below)



Concept/Principle of Locality

Locality model states that a Process migrates from one locality to another (w.r.t generated addresses) while it executes and localities may overlap.

- Program and data references within a process tend to cluster.
- Only a few pages/pieces of a process will be needed over a short period of time.
- Possible to make intelligent guesses about which pages/pieces will be needed in the future.
- This suggests that virtual memory may work efficiently.
- Locality of reference of a process refers to its most recent/active pages.

  Using working-set model Trashing can be prevented.

03.

Sol: When a page fault occurs, there is a trap to a OS to service it. Following are the steps performed for service:

1. Find a free frame in physical memory
2. Schedule a disk operation to read the desired page into newly allocated frame.
3. When the disk read is complete, modify the page table entry of newly brought page.
4. Restart the instruction that was interrupt due to page fault.

Chapter 10 File System Implementation

Objective Practice Solutions

01. Ans: (a)

Sol: Convert the Hex value to binary stream calculate the % of '0's to the total Binary length of string.

02. Ans: (a)

Sol: Surface capacity = 512 * 1K * 4 KB

$= 2^{31} = 2$ GB

$\therefore$ Disk capacity = 32 * surface capacity

$= 32 \times 2$ GB $= 64$ GB

03. Ans: (d)

Sol: All peripherals (External devices) require device driver.

05. Ans: (a)

Sol: Seek time is the dominating time in most of the IO operations.

09. Ans: (d)

Sol: Maximum file size

$= 8 \times 2KB + 1K \times 2KB + 1K \times 1K \times 2KB = 2.2$ GB

10. Ans: (b)

Sol: Number of Blocks on Disk $= \frac{40MB}{2\,KB} = 20\,K$

1 Block can store 16 K bits;

Hence 20 K bits can be stored in 2 Blocks.

12. Ans: (a)

Sol: $\frac{1024K}{4B} = 256\,K$

$256\,K \times 4 = 1024\,K = 1M$

13. Ans: (b)

Sol: **Mounting:** Before your computer can use any kind of storage device (HDD, CD-ROM) your OS must make it accessible through the computer's file system. This process is called mounting.

If mounting is automatically done by OS, then it is known as implicit mounting.

If mounting is done by user (line in Linux), it is known as explicit mounting.

Conventional Practice Solutions

01.

Sol: File with 2 copies can not reside in the same directory, so those are kept on two different locations.

Keeping two names of the same file can resolve the above problem as both files of different names can be kept in same directory.

02.

Sol: If important file system data are cached then before power off, computer system shutting down helps the cached data to be stored in disk and saved before power off.

Chapter 11 Protection & Security

Objective Practice Solutions

02. Ans: (c)

Sol: The unique name in capabilities can be reused, but only if it can be guaranteed that no references remain to the old use of the name.

Conventional Practice Solutions

01.

Sol:
1. **Trojan Horse:** A code segment the misuses its environment is called a Trojan horse.
2. **Trap door:** The designer of a program a system might leave a hole in the software that only he/she is capable of using. That is called as trap door.
3. **Warms:** It is a process that uses the spawn mechanism to clobber system performance. This worm spawns copies of itself using up system resources and perhaps locking out system use by all other processes.
4. **Viruses:** Like worms, viruses are designed to spread into other programs and can cause havoc in a system, including modifying or destroying files and causing system crashes and program malfunctions.
5. **Denial of service:** It does not involve gaining information or stealing resources, but rather disabling legitimate use of a system as facility.

Chapter 12 Introduction to Database

Objective Practice Solutions

01. Ans: (c)

Sol: DBMS is a collection of program that enables user to create and maintain a database.

02. Ans: (b)

Sol: Physical database design is the process of selecting the data storage and data access characteristics of the database.

03. Ans: (c)

Sol: A database is a collection of related data necessary to manage an organization.

04. Ans: (b)

Sol: The database environment has Users, Database and Database administration except Separate files.

05. Ans: (b)

Sol: In Relational model of database, data is stored in tables.

07. Ans: (a)

Sol: Data redundancy is not the feature of database.

08. Ans: (a)

Sol: Record stores the data items that are grouped together.

09. Ans: (d)

Sol: NULL indicates that there is no value.

10. Ans: (c)

Sol: Architecture of a database is viewed as three levels.

11. Ans: (d)

Sol: Tree is used to organize the records in hierarchical model.

13. Ans: (c)

Sol: Conceptual design involves modeling independent of the DBMS.

14. Ans: (d)

Sol: DBMS helps to achieve data independence and centralized control of data.

Chapter 13 ER & Relational Model

Objective Practice Solutions

01. Ans: (c)

Sol: If two students hold joint account then BankAccount_Num is not a candidate key and will not uniquely determine other attribute.

03. Ans: (b)

Sol: Derived attribute is an attribute that represents a value that is derivable from two or more attributes in an entity

04. Ans: (d)

Sol: On removal of row (2,4), row (5,2) and (7,2) must also be deleted as they depend on value 2. On removal of row (5,2), row (9,5) must also be deleted as it depends on value 5.

05. Ans: (b)

Sol: Entity type - Relation
Key attributes - Primary key
Composite attributes - Set of simple attributes
Weak Entity set - Child table
Value set - Domain

06. Ans: (b)

Sol: Name of the dependent : Discriminator attribute
Salary of a person: Composite attribute
Telephone number : Multi valued attribute
Date of birth: Stored attribute

07. Ans: (b)

Sol: When a tuple from the dominant is deleted the related tuple from the subordinate also be deleted.

08. Ans: (b)

Sol:

09. Ans: (c)

Sol: The tuples present at a particular moment is called relation instance.

10. Ans: (c)

Sol: Cardinality is the number of tuples in a relation instance.

11. Ans: (b)

Sol: In an E-R diagram, entities are represented by rectangles

12. Ans: (c)

Sol: In an E-R, diagram relationship is represented by diamond shaped box

13. Ans: (a)

Sol: Rows of a relation are called tuples

14. Ans: (a)

Sol: The employee salary should not be greater than Rs.2000, comes under integrity constraint

15. Ans: (d)

Sol: Students and courses enrolled is an example of many-to-many relationship.

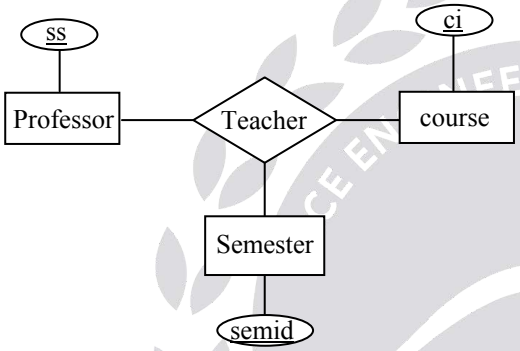
16. Ans: (a)

Sol: An attribute of one table matching the primary key of another table, is called as foreign key.

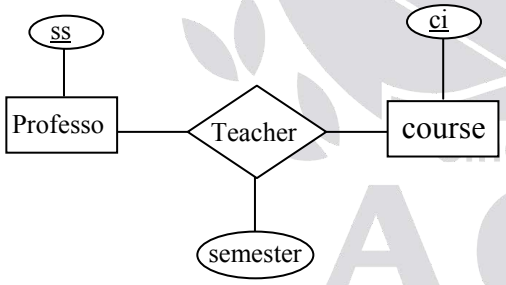
17. Ans:

Sol:

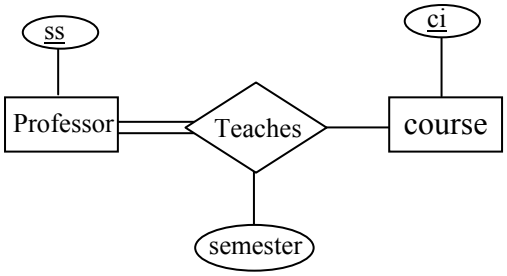
1



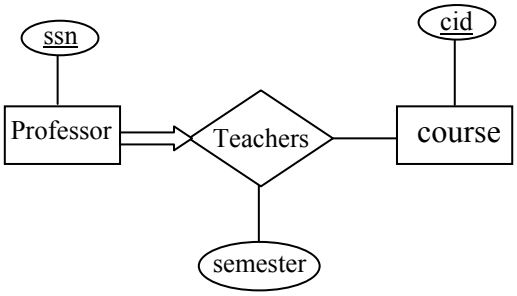
2



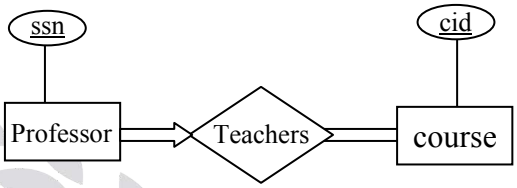
3



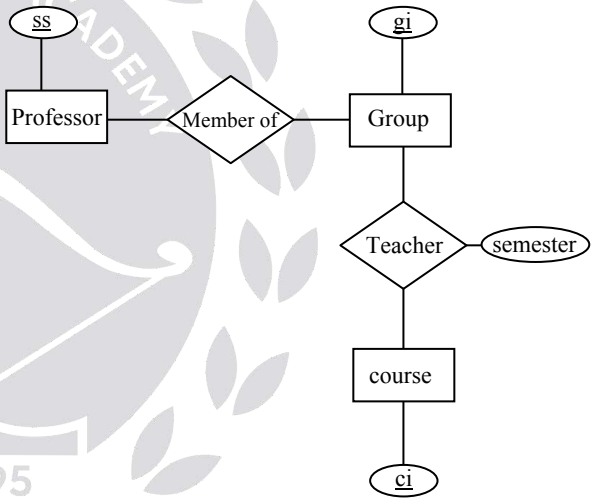
4



5



6

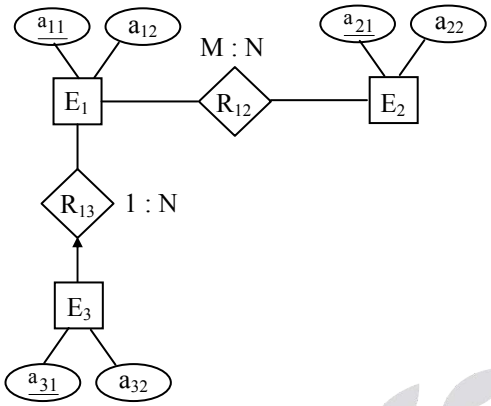


18. Ans: (b)

Sol: Strong entities E_1 and E_2 are represented as separate tables, in addition to that many to many relationship (R_2) must be converted as separate table by having primary keys of E_1 and E_2 as foreign keys. One to many relationship must be transferred to 'many' side table by having primary key of one side as foreign key. Hence we will have minimum of 3 tables.

19. Ans: (c)

Sol:



If we convert into Relational model, the following relations will result.

1) $E_1(\underline{a_{11}}, a_{12})$

2) $E_2(\underline{a_{21}}, a_{22})$

3) $R_{12}(\underline{a_{11}\, a_{21}})$

4) $E_3R_{13}(\underline{a_{31}}, a_{32}, a_{11})$

20. Ans: (b)

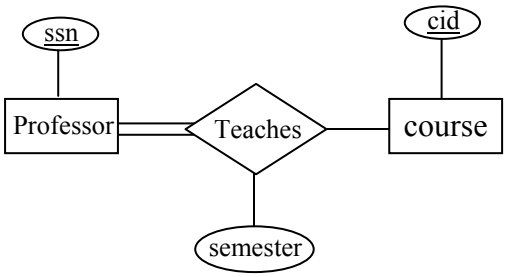
Sol: Strong entities E_1 and E_2 are represented as separate tables, in addition to that many to many relationship (R_2) must be converted as separate table by having primary keys of E_1 and E_2 as foreign keys. One to many relationship must be transferred to 'many' side table by having primary key of one side as foreign key. Hence we will have minimum of 3 tables.

Conventional Practice Solutions

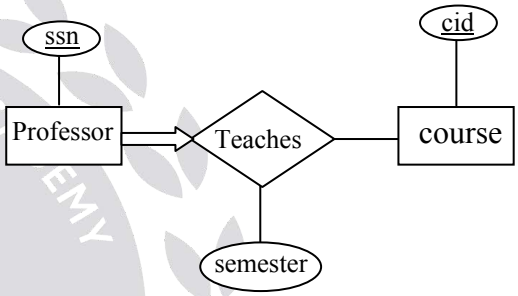
01.

Sol:

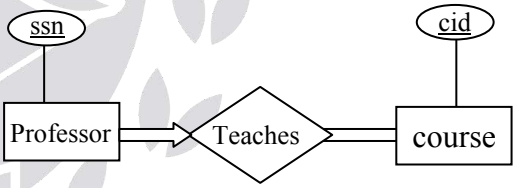
1



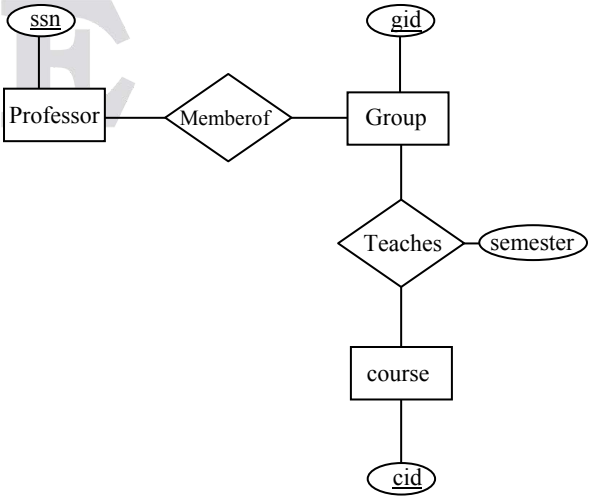
2



3

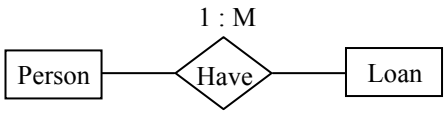


4



02. Ans: (c)

Sol:



In Relational mode we represent the above situation with only two relations person and (Loan, Have).

03. Ans: 3

Sol: Strong entities E1 and E2 are converted as separate tables. Since A23 is a multi valued attribute it should also be converted as separate table. Relationship R is transferred to 'm' side (E2).

Chapter 14 Functional Dependencies

Objective Practice Solutions

01. Ans: (b)

Sol: From the given schema we conclude that "A functionally determines B and B does not functionally determine C'.

02. Ans: (d)

Sol: The FD BC → A holds good

03. Ans: (d)

Sol: The FD C → B doesn't hold over Relation R.

04. Ans: (d)

Sol: The FD BC → A holds good.

06. Ans: (c)

Sol: Trivial FDs are satisfied by all relations in a Database

08. Ans: (c)

Sol: AF^+ = AFDE not ACDEFG as given.

09. Ans: (d)

Sol: AB^+ = {ABCDEF}

10. Ans: (b)

Sol: CD^+ from functional dependencies

(FDs) = **C**DE**A**B, it includes RHS attributes AC so it can be derived from FDs BD^+ from functional dependencies

(FDs) = BD only, RHS attributes CD are not included in the closure hence it cannot be derived BC^+ from functional dependencies

(FDs) = B**CD**EA, it includes RHS attributes CD, so it can be derived from FDs AC^+ from functional dependencies

(FDs) = A**CB**DE, it includes RHS attributes BC so it can be derived from FDs

11. Ans: (b)

Sol: $ACEH^+$ contains all the attributes of R.

12. Ans: (d)

Sol: Closure of AEH^+ = BEH^+= DEH^+= A, B, C, D, E, H. If any closure includes all attributes of a table then it can become candidate key of the table. Closure of AEH, BEH, DEH includes all attributes of table hence they are candidate keys.

13.

Sol: CK: ACD, BCD, ECD.

14. Ans: (d)

Sol: ck: ABF, EBF, CBF, ADF, EDF, CDF.

16. Ans: (a)

Sol: G not covers the dependency D → E of F.

18.

Sol: $\left.\begin{array}{l} AD \rightarrow CF \\ C \rightarrow B \\ B \rightarrow E \end{array}\right\}$ Canonical set

$AD \rightarrow C$
$C \rightarrow B$
$AB \rightarrow F$
$B \rightarrow E$
} Minimal set

19.

Sol: In $AB \rightarrow C$; B is redundant as A determines B and in $D \rightarrow AC$; C is redundant as D determines C. Therefore the minimal set: $A \rightarrow BC$ and $D \rightarrow AC$.

Conventional Practice Solutions

01.

Sol: $CF^+ = \{C, F, G, E, A, D\}$
$BG^+ = \{B, G, A, C, D\}$
$AF^+ = \{A, F, E, D\}$
$AB^+ = \{A, B, C, D, G\}$

02.

Sol: $AC^+ = \{ABCDE\}$

03.

Sol: CK: AB.

Chapter 15 Normalizations

Objective Practice Solutions

01.

Sol: (1) $C \rightarrow D$

$C \rightarrow A$

$B \rightarrow C$

C.K: B, 2NF but not 3NF

(2) 2NF but not 3NF as no partial dependency CK: BD.

(3) R is in 3NF but not in BCNF

R $\begin{cases} D^+ = \{D, A\}\ R_1 \\ \{DBC\}\ R_2 \end{cases}$

Not D.P ABC $\rightarrow$ D lost

(4) C.K = A

No Partial Dependency

$\therefore$ 2NF

R is in 2NF but not in BCNF & 3NF

$BC^+ = \{BCD\}\ R_1$

$= \{ABC\}\ R_2$

(5) $AB \rightarrow C$

$AB \rightarrow D$

$C \rightarrow A$

$D \rightarrow B$

Candidate Keys = AB, CD, BC, AD

R is in 3NF but not in BCNF.

C+ $\{C, A\}\ R_1$

D+ $\{D, B\}\ R_2$

$\{C, D\}\ R_3$

Not D.P AB $\rightarrow$ CD lost.

02. Ans: (d)

Sol: Since $R1 \cap R2 \neq \phi$, R1 and R2 might have two attributes each and no common attribute. Any relation with 2 attributes satisfies 2 NF and 3 NF.

03. Ans: (b)

Sol: 1 represents Partial FD and 2 represents T.D

04. Ans: (c)

Sol: In FD $A \rightarrow C$, R in 1NF but not in 2NF

05. Ans: (a)

Sol: In FD $D \rightarrow C$, R is in 2NF but is in 3NF

06. Ans: (a)

Sol: Every binary relation is in BCNF.

08. Ans: (a)

Sol: Relation contains PFD: $B \rightarrow F$.

09. Ans: (b)

Sol: Relation contains TD: $C \rightarrow E$.

10. Ans: (d)

Sol: $BC \rightarrow D$ is Transitive FD.

11. Ans: (a)

Sol: B → F is PFD.

12. Ans: (d)

Sol: 1 NF - Unique rows

2 NF - Partial Dependencies

BCNF - Candidate keys

3 NF - Transitive dependencies

13. Ans: (b)

Sol: ACH is a key

14. Ans: (d)

Sol: E → G is Partial F.D.

15. Ans: (b)

Sol: In 2 NF every non-prime attribute is fully dependent functionally on the candidate key of relational schema

16.

Sol: R_1(CD)

R_2(FG)

R_3(AF)

R_4(ABCE)

Conventional Practice Solutions

01.

Sol: AB^+ = ABCDE

C^+ = CDE

A^+= AE

So Key is AB.

From A → E, attribute A is part of key so it is in 1NF.

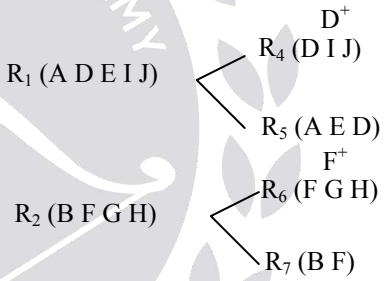
02.

Sol: R is in 1NF ∴ decompose to 2NF

A^+ = {A, D, E, I, J} R_1 = 2NF

B^+ = {B, F, G, H} R_2 = 2NF

{A, B, C} R_3 BCNF



Then decompose into 2NF

R_1 (ADEIJJ)

R_2 (BFGH)

R_3 (ABC)

3NF also in BCNF

R_3 (ABC)

R_4 (DIJ)

R_5 (AED)

R_6 (FGH)

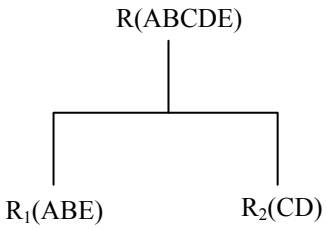
R_7 (BF)

AB^+ is key.

03.

Sol: Given relation R(ABCDE) with FD's are {B → E, C → D, A → B}. The candidate key for R is {AC}

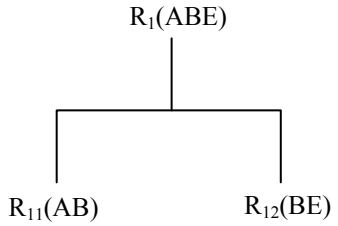
∴ The relation R is not in 2NF



But the decomposition relation of R into R_1 and R_2 are lossy decomposition. To avoid lossy decomposition, we add one more relation R_3(AC)

∴ Total relation in 2NF is R_1(ABE), R_2(CD), R_3(AC)

But the relation R_1 is not in 3NF because attribute E is transitively depended on A. Therefore relation R_1 is not in 3NF. Relation R_1 is decomposed into



∴ Total number of relation in 3NF is R_{11}(AB), R_{12}(BE), R_2(CD), R_3(AC).

Chapter 16 Transaction & Concurrency Control

Objective Practice Solutions

01. Ans: (c)

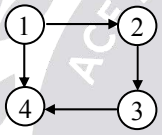
Sol: Atomocity : Recovery Manager
Isolation : Concurrency Control Sub system
Durability: Transaction Manager
Consistency Preservation : User

02. Ans: (d)

Sol: Every strict schedule is both recoverable and cascadeless.

03. Ans: (c)

Sol: Precedence graph



04. Ans: (b)

Sol: In S_1 $R_2(x)$ and $W_1(x)$ conflicts hence $T_2 < T_1$. Also $W_1(y)$ and $W_2(y)$ conflicts hence $T_1 < T_2$. There is no serial schedule that satisfies both $T_2 < T_1$ and $T_1 < T_2$.

In S_4 $R_2(x)$ and $W_1(x)$ conflicts hence $T_2 < T_1$. Also $W_1(y)$ and $W_2(y)$ conflicts hence $T_1 < T_2$. There is no serial schedule that satisfies both $T_2 < T_1$ and $T_1 < T_2$

05. Ans: (d)

Sol: S1 and S2 are conflict equivalent to serial schedule T_2, T_3, T_1.

S3 is not conflict equivalent as 2RA, 3WA ($T_2 < T_3$) and 3WA, 2WA ($T_3 < T_2$) are the conflict operations. There is no serial schedule that satisfies both $T_2 < T_3$ and $T_3 < T_2$.

06.

Sol: Not Conflict Serializable,
Not View Serializable,
Recoverable, Avoids Cascading aborts, Not strict.

07.

Sol: Conflict Serializable,
View Serializable,
Serializable,
Recoverable,
Cascading aborts, Not strict

08.

Sol: Conflict Serializable,
View serializable,
Serializable,
Recoverable,
Cascading aborts, Not strict

09.

Sol: Not Conflict Serializable,
Not View Serializable,
Not avoids cascading aborts, not strict, Recoverable, cascading aborts

10.

Sol: Conflict Serializable,
View serializable,
Serializable,
Recoverable,
Avoids cascading aborts,
Not strict

11.

Sol: Not Conflict Serializable, Views serializable through Thomas write rule, Serializable, Recoverable, Avoids cascading aborts, Not strict

12.

Sol: Conflict Serializable,
View Serializable,
Serializable,
Recoverable,
Avoids cascading aborts,
Not strict

13.

Sol: Not Conflict Serializable,
Not View Serializable,
Not Serializable,
Not Recoverable,
No need cascading aborts,
Not strict

14.

Sol: Conflict Serializable,
View Serializable,
Serializable,
Not Recoverable,
No need cascading aborts,
Not strict

15.

Sol: Conflict Serializable,
View Serializable,
Serializable,
Recoverable,
Avoid cascading aborts,
Strict

16.

Sol: Conflict Serializable,
View Serializable,
Serializable,
Recoverable,
Avoid cascading aborts,
Strict

17.

Sol: Not Conflict Serializable,
View Serializable as per Thomas write rule,
Serializable,
Recoverable,
Cascading aborts,
Not strict

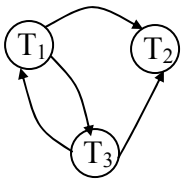
Conventional Practice Solutions

01.

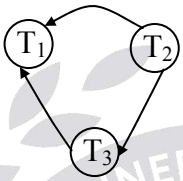
Sol: (i) Conflict serializable

$T_3 \rightarrow T_1 \rightarrow T_2$

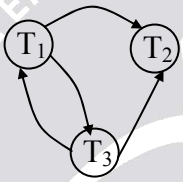
(ii) Not Conflict serializable



(iii) Conflict serializable



(iv) Not Conflict serializable

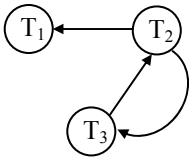


02.

Sol:

T_1	T_2	T_3
	R(A)	
		R(B)
	W(A)	
R(A)		
		W(B)
	R(B)	
W(A)		
	W(B)	
	R(C)	
	W(C)	
R(C)		
W(C)		
		W(B)

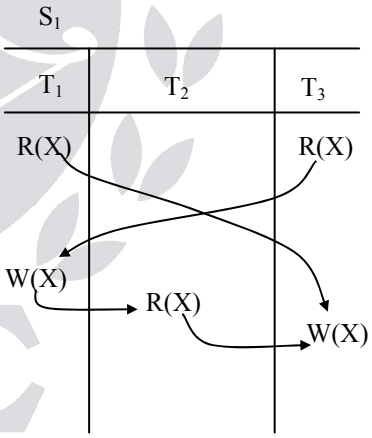
Hence ① shows that the transaction T_2 reads the data item which has previously written by transaction T_3 ② shows that the transaction T_3 writes the data item which has previously written by T_2.

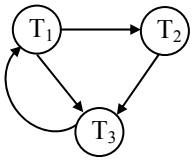


As there is presence of cycle the given schedule is not serializable.

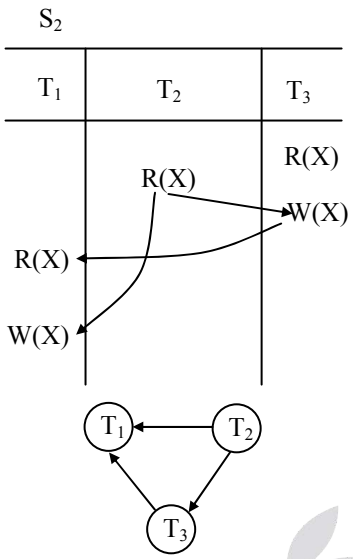
03.

Sol:





Here cycle is present so it is not serializable



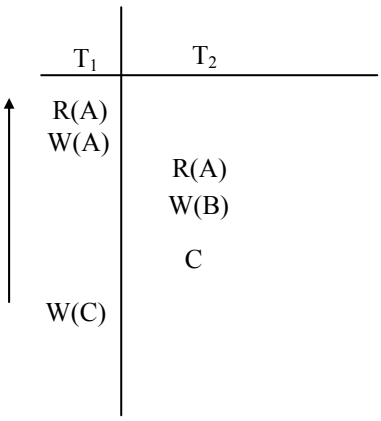
Here, there is no presence of cycle.

So it is also serializable.

Serializable but not S_1.

04. Ans: (c)

Sol:



Even though transaction T_1 is roll backed, T_2 is committed before T_1. So the total schedule is non recoverable schedule.