NIMCET Computer Awareness Sample Paper-15

Duration: 15 Minutes **Maximum Marks: 120**

Instructions

- This paper contains **20** Multiple Choice Questions (Single Correct).
- Each correct answer carries **+6 marks**.
- Each incorrect answer carries: **−1.5** marks.
- Unattempted questions carry **0** marks.
- Only one option is correct for each question.
- Use of mobile phones, smartwatches, calculators, or any electronic gadgets is strictly prohibited.

Q1. A computer system has a 32 KB direct-mapped cache with a block size of 32 bytes. The processor uses 32-bit physical addresses.

Suppose a memory block has address 0x1234ABCD. Which of the following blocks will definitely map to the same cache line as this block?

(A) 0x1234BBCD

(B) 0x1235ABCD

(C) 0x1254ABCD

(D) 0x2234ABCD

Q2. A processor has a base CPI of 1.2. Memory instructions constitute 30% of all instructions. The cache miss rate is 4%, and each miss incurs a penalty of 25 clock cycles.

What is the effective CPI of the processor?

(A) 1.30

(B) 1.45

(C) 1.50

(D) 1.65

Q3. A 4-stage instruction pipeline has stage delays of 5 ns, 7 ns, 8 ns and 6 ns respectively.

Ignoring pipeline register overhead, what is the clock period of the pipelined processor?

(A) 5 ns

(B) 6 ns

(C) 7 ns

(D) 8 ns

Q4. A hard disk rotates at 12000 RPM.

Assuming negligible seek time, what is the average rotational latency?

(A) 1.5 ms

(B) 2.5 ms

(C) 3 ms

(D) 5 ms

Q5. The decimal number

$$-101$$

is represented using 8-bit two's complement arithmetic.

What is its binary representation?

(A) 10011011

(B) 10011101

(C) 10010111

(D) 10100101

Q6. The hexadecimal number

$$(FE7A)_{16}$$

is converted into decimal form.

What is the resulting decimal value?

(A) 65146

(B) 65178

(C) 65114

(D) 65210

Q7. A Hamming code uses 6 parity bits.

What is the maximum number of data bits that can be protected while still allowing single-bit error correction?

(A) 57

(B) 58

(C) 59

(D) 60

Q8. The binary number

$$11110000$$

is converted into Gray code.

Which of the following is obtained?

(A) 10001000

(B) 10010000

(C) 10100000

(D) 10000000

Q9. In IEEE-754 single precision format, the bit pattern corresponds to a normalized floating-point number whose exponent field is

$$10000110_2.$$

What is the actual exponent?

(A) 5

(B) 6

(C) 7

(D) 8

Q10. A 14-bit Analog-to-Digital Converter is used in a measurement system.

How many distinct quantization levels are available?

(A) 8192

(B) 12288

(C) 16384

(D) 32768

Q11. A virtual memory system uses 8 KB pages and 36-bit virtual addresses.

How many virtual pages are present in the address space?

(A) 2^{20}

(B) 2^{21}

(C) 2^{22}

(D) 2^{23}

Q12. A page table contains 2^{20} entries. Each entry occupies 8 bytes.

What is the total size of the page table?

(A) 4 MB

(B) 8 MB

(C) 16 MB

(D) 32 MB

Q13. A cache has a hit ratio of 98%. Cache access time is 1 ns and main memory access time is 100 ns.

Assuming memory is accessed only on cache misses, what is the effective memory access time?

(A) 2 ns

(B) 3 ns

(C) 4 ns

(D) 5 ns

Q14. Consider a RAID-5 system consisting of five identical disks, each of capacity 2 TB.

Ignoring filesystem overhead, what is the usable storage capacity available to the user?

(A) 6 TB

(B) 8 TB

(C) 10 TB

(D) 12 TB

Q15. Using Boolean algebra, simplify the expression

$$AB + \overline{A}B + A\overline{B} + \overline{A}\overline{B}.$$

The simplified form is:

(A) A

(B) B

(C) 1

(D) 0

Q16. How many prime implicants are present in the Boolean function

$$F(A, B) = \Sigma m(0, 1, 2, 3)?$$

Assume the function is represented using a Karnaugh map.

(A) 0

(B) 1

(C) 2

(D) 4

Q17. A Boolean function has 6 variables and exactly 20 minterms in its canonical Sum-of-Minterms representation.

How many input combinations make the function evaluate to 0?

(A) 20

(B) 32

(C) 44

(D) 64

Q18. An IPv4 network uses the subnet mask

$$255.255.255.240.$$

How many subnets are obtained when a Class C network is subnetted using this mask?

(A) 8

(B) 14

(C) 16

(D) 32

Q19. Consider the relation

$$R(A, B, C, D, E)$$

with functional dependencies

$$A \to BC, \quad C \to D, \quad D \to E.$$

If A is a candidate key, which attribute is transitively dependent on A?

(A) B

(B) C

(C) D

(D) A

Q20. In the Banker's Algorithm, the Need matrix is computed from two matrices named Allocation and Maximum.

Which of the following expressions correctly defines the Need matrix?

(A) Need = Allocation + Maximum

(B) Need = Maximum - Allocation

(C) Need = Allocation - Maximum

(D) Need = Available - Allocation

Detailed Solutions

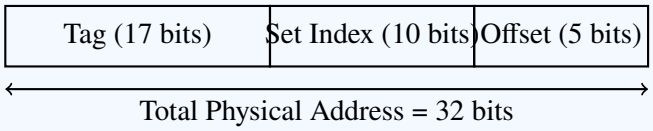
Q1.

Solution

Concept: In a direct-mapped cache, an address is divided into:

- **Block Offset:** Identifies a byte within a block.
- **Set Index:** Identifies the cache line.
- **Tag:** Identifies the memory block.

Two addresses map to the same cache line if they have the same Set Index bits; differences may occur only in the Tag or Offset fields.



Solution: Step 1: Determine the address fields:

- Block Offset = $\log_2(32) = 5$ bits
- Cache Lines = $\frac{32\text{ KB}}{32\text{ B}} = 2^{10}$
- Set Index = $\log_2(2^{10}) = 10$ bits
- Tag = $32 - 10 - 5 = 17$ bits

Thus, bits 5–14 form the Set Index, while bits 15–31 form the Tag.

Step 2: Compare the addresses:

- `0x1234BBCD` changes bits belonging to the Set Index, so it maps to a different cache line.
- `0x1235ABCD` changes only Tag bits, leaving the Set Index unchanged.

Step 3: Verify:

$$0x1235ABCD - 0x1234ABCD = 0x10000 = 65{,}536 \text{ Bytes}$$

Since $65{,}536 = 2 \times 32$ KB, the address maps to the same cache line.

Final Answer: 0x1235ABCD

Answer: (B)

Go Back to Question 1

Q2.

Solution

Concept: The effective Cycles Per Instruction (CPI) of a processor with cache memory accounts for stall cycles caused by memory access misses (miss penalty). It is calculated as:

$$\text{Effective CPI} = \text{Base CPI} + \text{Memory Stall CPI}$$

where:

$$\text{Memory Stall CPI} = \text{Fraction of Memory Instructions} \times \text{Cache Miss Rate} \times \text{Miss Penalty}$$

Solution: Step 1: Identify the given workload and hardware parameters:

- Base CPI = 1.2
- Fraction of memory instructions = 30% = 0.30
- Cache miss rate = 4% = 0.04
- Miss penalty = 25 clock cycles

Step 2: Calculate the memory stall cycles per instruction:

$$\begin{aligned}\text{Memory Stall CPI} &= 0.30 \times 0.04 \times 25 \\ &= 0.30 \times 1.0 \\ &= 0.30 \text{ stall cycles}\end{aligned}$$

Step 3: Add the stall cycles to the base CPI to find the effective CPI:

$$\begin{aligned}\text{Effective CPI} &= \text{Base CPI} + \text{Memory Stall CPI} \\ &= 1.2 + 0.30 \\ &= 1.50\end{aligned}$$

Thus, the effective CPI of the processor is 1.50. Option C is correct.

Final Answer: 1.50

Answer: (C)

Go Back to Question 2

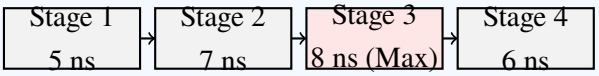
Q3.

Solution

Concept: In a pipelined processor, instruction execution is split into multiple independent stages. Because all stages must operate synchronously under a single global clock signal:

- The clock period (T_{clk}) must be long enough to allow the slowest stage in the pipeline to fully complete its operations.
- Ignoring pipeline register overhead (setup, propagation delay, and skew), the minimum clock period is bounded by the maximum delay among all pipeline stages:

$$T_{\text{clk}} = \max(\text{Stage}_1, \text{Stage}_2, \ldots, \text{Stage}_n)$$



Solution: Step 1: Identify the individual stage delays:

- Stage 1 delay = 5 ns
- Stage 2 delay = 7 ns
- Stage 3 delay = 8 ns
- Stage 4 delay = 6 ns

Step 2: Find the maximum stage delay:

$$\begin{aligned} T_{\text{clk}} &= \max(5 \text{ ns}, 7 \text{ ns}, 8 \text{ ns}, 6 \text{ ns}) \\ &= 8 \text{ ns} \end{aligned}$$

The minimum clock period required for the pipelined processor is 8 ns. Options A, B, and C are too short to accommodate the slowest stage (Stage 3). Thus, Option D is correct.

Final Answer: 8 ns

Answer: (D)

Go Back to Question 3

Q4.

Solution

Concept: The rotational latency of a hard disk is the time required for the target sector on a platter track to rotate underneath the read/write head.

- The time for one full 360° rotation (T_{rot}) depends on the disk's rotational speed, typically expressed in Rotations Per Minute (RPM).
- On average, the read/write head must wait for half of a full rotation to find the target sector. Thus, average rotational latency is defined as:

$$\text{Average Rotational Latency} = \frac{T_{\text{rot}}}{2} = \frac{1}{2 \times \text{Rotational Speed}}$$

Solution: Step 1: Convert the rotational speed from minutes to seconds:

$$\begin{aligned}\text{Rotational Speed} &= 12,000 \text{ RPM} \\ &= \frac{12,000 \text{ rotations}}{60 \text{ seconds}} = 200 \text{ rotations per second (Hz)}\end{aligned}$$

Step 2: Calculate the time required for one full rotation (T_{rot}):

$$\begin{aligned}T_{\text{rot}} &= \frac{1}{200 \text{ rotations/second}} \\ &= 0.005 \text{ seconds} = 5 \text{ ms}\end{aligned}$$

Step 3: Calculate the average rotational latency:

$$\begin{aligned}\text{Average Rotational Latency} &= \frac{T_{\text{rot}}}{2} \\ &= \frac{5 \text{ ms}}{2} = 2.5 \text{ ms}\end{aligned}$$

Thus, the average rotational latency is 2.5 ms. Option B is correct.

Final Answer: 2.5 ms

Answer: (B)

Go Back to Question 4

Q5.

Solution

Concept: In an n-bit two's complement binary system, a negative integer $-X$ is represented by taking the two's complement of its positive counterpart $+X$. The procedure is:

(a) Write the binary representation of the positive value $+X$ using n bits.

(b) Perform bitwise negation (one's complement) on $+X$.

(c) Add 1 to the negated result to obtain the final two's complement representation.

Solution: Step 1: Write down +101 as an 8-bit unsigned binary number:

$$\begin{aligned} 101 &= 64 + 32 + 4 + 1 \\ &= 2^6 + 2^5 + 2^2 + 2^0 \\ \text{Binary representation} &= 01100101_2 \end{aligned}$$

Step 2: Invert all bits to find the one's complement of +101:

$$\begin{aligned} \text{One's Complement} &= \overline{01100101}_2 \\ &= 10011010_2 \end{aligned}$$

Step 3: Add 1 to get the two's complement representation of −101:

$$\begin{aligned} \text{Two's Complement} &= 10011010_2 + 00000001_2 \\ &= 10011011_2 \end{aligned}$$

Step 4: Verify the result by expanding back to decimal:

$$\begin{aligned} \text{Value} &= -128 \cdot 2^7 + 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^1 + 1 \cdot 2^0 \\ &= -128 + 16 + 8 + 2 + 1 \\ &= -128 + 27 = -101 \end{aligned}$$

This confirms that '10011011' is correct, matching Option A.

Final Answer: 10011011

Answer: (A)

Go Back to Question 5

Q6.

Solution

Concept: A hexadecimal (base-16) number is converted to its decimal (base-10) equivalent by summing each digit multiplied by its corresponding positional power of 16:

$$(d_3d_2d_1d_0)_{16} = d_3 \cdot 16^3 + d_2 \cdot 16^2 + d_1 \cdot 16^1 + d_0 \cdot 16^0$$

where the hexadecimal digit characters map to decimal values as: A = 10, B = 11, C = 12, D = 13, E = 14, F = 15.

Solution: Step 1: Write down the digits of $(FE7A)_{16}$ and translate them into decimal values:

- F $\rightarrow$ 15
- E $\rightarrow$ 14
- 7 $\rightarrow$ 7
- A $\rightarrow$ 10

Step 2: Apply the positional weights of base-16:

$$\begin{aligned}\text{Decimal Value} &= 15 \cdot 16^3 + 14 \cdot 16^2 + 7 \cdot 16^1 + 10 \cdot 16^0 \\ &= 15 \cdot 4096 + 14 \cdot 256 + 7 \cdot 16 + 10 \cdot 1\end{aligned}$$

Step 3: Compute the individual products and sum them:

$$\begin{aligned}15 \cdot 4096 &= 61,440 \\ 14 \cdot 256 &= 3,584 \\ 7 \cdot 16 &= 112 \\ 10 \cdot 1 &= 10 \\ \text{Total} &= 61,440 + 3,584 + 112 + 10 \\ &= 65,024 + 112 + 10 \\ &= 65,136 + 10 = 65,146\end{aligned}$$

Thus, $(FE7A)_{16}$ converts to 65146. Option A is correct.

Final Answer: 65146

Answer: (A)

Go Back to Question 6

Q7.

Solution

Concept: For single-bit error correction (SEC) using Hamming code, the number of parity bits p and data bits d must satisfy the inequality:

$$2^p \geq d + p + 1$$

To find the maximum number of data bits that can be protected by a fixed number of parity bits p, we rearrange the inequality to solve for d:

$$d \leq 2^p - p - 1$$

Solution: Step 1: Identify the given number of parity bits:

$$p = 6 \text{ bits}$$

Step 2: Substitute $p = 6$ into the rearranged Hamming inequality:

$$d \leq 2^6 - 6 - 1$$
$$d \leq 64 - 7$$
$$d \leq 57$$

Step 3: Extract the maximum value: The maximum integer value of d that satisfies the inequality is 57. Thus, 6 parity bits can protect a maximum of 57 data bits in a $(63, 57)$ Hamming codeword. Option A is correct.

Final Answer: $\boxed{57}$

Answer: (A)

Go Back to Question 7

Q8.

Solution

Concept: To convert binary B to Gray code G:

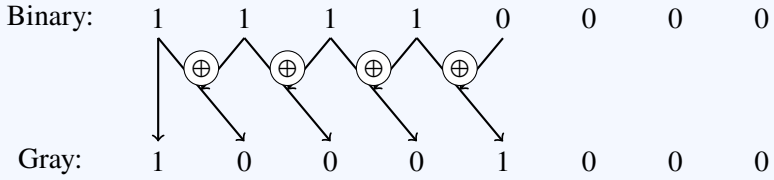
- The MSB remains unchanged:

$$g_{n-1} = b_{n-1}$$

- Each remaining Gray bit is obtained using XOR:

$$g_i = b_i \oplus b_{i+1}$$

Thus, each Gray bit is formed by XORing two adjacent binary bits.



Solution:

Step 1: For 11110000_2, compute the Gray code bits:

$$g_7 = 1$$
$$g_6 = 1 \oplus 1 = 0$$
$$g_5 = 1 \oplus 1 = 0$$
$$g_4 = 1 \oplus 1 = 0$$
$$g_3 = 1 \oplus 0 = 1$$
$$g_2 = 0 \oplus 0 = 0$$
$$g_1 = 0 \oplus 0 = 0$$
$$g_0 = 0 \oplus 0 = 0$$

Step 2: Combining the bits:

$$G = 10001000_2$$

Final Answer: $\boxed{10001000}$

Answer: (A)

Go Back to Question 8

Q9.

Solution

Concept: Under the IEEE-754 single-precision standard:

- The exponent is stored as an 8-bit unsigned value in a biased form.
- The bias added to any single-precision exponent is 127.
- To calculate the actual exponent (E), we subtract the bias from the decimal equivalent of the biased exponent field (E_{biased}):

$$E = E_{\text{biased}} - 127$$

Solution: Step 1: Convert the binary exponent field value 10000110_2 to decimal:

$$\begin{aligned} E_{\text{biased}} &= (1 \cdot 2^7) + (0 \cdot 2^6) + (0 \cdot 2^5) + (0 \cdot 2^4) + (0 \cdot 2^3) + (1 \cdot 2^2) + (1 \cdot 2^1) + (0 \cdot 2^0) \\ &= 128 + 0 + 0 + 0 + 0 + 4 + 2 + 0 \\ &= 134 \end{aligned}$$

Step 2: Apply the bias subtraction equation:

$$\begin{aligned} E &= E_{\text{biased}} - 127 \\ &= 134 - 127 \\ &= 7 \end{aligned}$$

The actual exponent represented by the field is 7. This matches Option C.

Final Answer: $\boxed{7}$

Answer: (C)

Go Back to Question 9

Q10.

Solution

Concept: An Analog-to-Digital Converter (ADC) resolves a continuous analog voltage range into discrete digital values. The precision is governed by the ADC resolution, expressed in bits (n). The total number of distinct quantization levels that can be generated is:

$$\text{Quantization Levels} = 2^n$$

Solution: Step 1: Identify the resolution of the ADC:

$$n = 14 \text{ bits}$$

Step 2: Calculate the number of quantization levels:

$$\text{Levels} = 2^{14}$$

Step 3: Evaluate the exponent:

$$2^{14} = 16,384$$

Thus, a 14-bit ADC produces 16384 distinct quantization levels. Options A (8192), B (12288), and D (32768) are incorrect.

Final Answer: 16384

Answer: (C)

Go Back to Question 10

Q11.

Solution

Concept: In a virtual memory system, the virtual address space is divided into two primary parts:

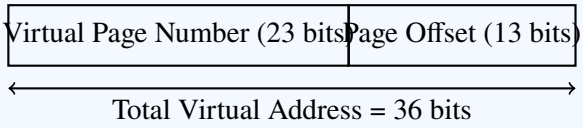
- **Virtual Page Number (VPN):** Identifies the specific virtual page referenced by the address.
- **Page Offset:** Identifies the specific byte within that virtual page.

The number of bits used for the page offset (d) is determined by the size of each page (S_{page}):

$$2^d = S_{\text{page}} \implies d = \log_2(S_{\text{page}})$$

The remaining bits in the virtual address (VA) represent the Virtual Page Number (VPN). The total number of virtual pages (N) present in the virtual address space is:

$$N = 2^{\text{VPN Bits}} = 2^{VA-d}$$



Solution: Step 1: Convert the page size from kilobytes to bytes to find the page offset bits (d):

$$\begin{aligned} S_{\text{page}} &= 8 \text{ KB} \\ &= 8 \times 1024 \text{ Bytes} \\ &= 2^3 \times 2^{10} \text{ Bytes} = 2^{13} \text{ Bytes} \\ \text{Page Offset Bits } (d) &= \log_2(2^{13}) = 13 \text{ bits} \end{aligned}$$

Step 2: Subtract the page offset bits from the total virtual address bits (36 bits) to find the bits allocated to the Virtual Page Number (VPN):

$$\begin{aligned} \text{VPN Bits} &= 36 \text{ bits} - 13 \text{ bits} \\ &= 23 \text{ bits} \end{aligned}$$

Step 3: Calculate the total number of virtual pages: Since 23 bits are allocated to the VPN, the address space can accommodate:

$$\text{Number of Virtual Pages} = 2^{23}$$

Thus, 2^{23} virtual pages are present in the address space. Option D is correct.

Final Answer: $\boxed{2^{23}}$

Answer: (D)

Go Back to Question 11

Q12.

Solution

Concept: A page table contains translations for virtual-to-physical mappings, where each virtual page has a corresponding Page Table Entry (PTE). The total memory storage required by a page table is computed by multiplying the total number of entries by the size of each entry:

$$\text{Total Page Table Size} = \text{Number of Entries} \times \text{Size of Each Entry}$$

To express the result in standard units, recall that:

$$1 \text{ MB} = 2^{20} \text{ Bytes} = 1,048,576 \text{ Bytes}$$

Solution: Step 1: Identify the given table parameters:

- Number of entries = 2^{20}
- Size of each entry = 8 Bytes

Step 2: Calculate the total storage size in bytes:

$$\begin{aligned} \text{Total Size} &= 2^{20} \times 8 \text{ Bytes} \\ &= 8 \times 2^{20} \text{ Bytes} \end{aligned}$$

Step 3: Convert the byte capacity into Megabytes (MB):

$$\begin{aligned} \text{Total Size} &= 8 \times \left(2^{20} \text{ Bytes}\right) \\ &= 8 \text{ MB} \end{aligned}$$

Thus, the total size of the page table is 8 MB, matching Option B. Options A (4 MB), C (16 MB), and D (32 MB) are incorrect.

Final Answer: 8 MB

Answer: (B)

Go Back to Question 12

Q13.

Solution

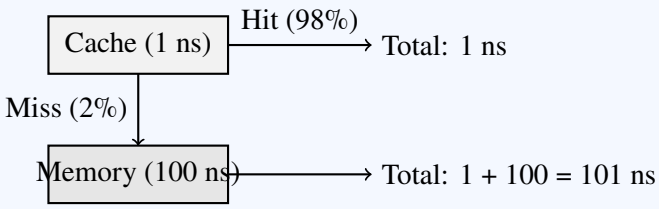
Concept: In memory subsystems, the effective memory access time (T_{eff}) is the average time required to perform a memory access, accounting for the distribution of cache hits and misses.

When main memory is accessed only on cache misses, the cache is checked first.

- On a cache hit (occurring with probability h), access takes T_{cache} time.
- On a cache miss (occurring with probability $1 - h$), the cache check still takes T_{cache} time, after which the main memory is accessed, taking an additional T_{mem} time.

The formula for the effective memory access time is:

$$\begin{aligned} T_{\text{eff}} &= h \cdot T_{\text{cache}} + (1 - h) \cdot (T_{\text{cache}} + T_{\text{mem}}) \\ &= T_{\text{cache}} + (1 - h) \cdot T_{\text{mem}} \end{aligned}$$



Solution: Step 1: Identify the given system parameters:

- Hit ratio (h) = 98% = 0.98
- Miss ratio ($1 - h$) = 2% = 0.02
- Cache access time (T_{cache}) = 1 ns
- Main memory access time (T_{mem}) = 100 ns

Step 2: Apply the values to the sequential effective access time formula:

$$\begin{aligned} T_{\text{eff}} &= T_{\text{cache}} + (1 - h) \cdot T_{\text{mem}} \\ &= 1 \text{ ns} + (0.02 \times 100 \text{ ns}) \\ &= 1 \text{ ns} + 2 \text{ ns} \\ &= 3 \text{ ns} \end{aligned}$$

Thus, the effective memory access time is 3 ns. Option B is correct.

Final Answer: 3 ns

Answer: (B)

Go Back to Question 13

Q14.

Solution

Concept: RAID-5 (Redundant Array of Independent Disks Level 5) uses block-level striping with distributed parity.

- Parity information is distributed across all disks in the array.
- The storage capacity equivalent to exactly one disk is consumed by parity data [14].
- Therefore, for an array of N identical disks each of capacity C, the usable storage capacity (C_{usable}) available for user data is:

$$C_{\text{usable}} = (N - 1) \times C$$

Solution: Step 1: Identify the given RAID parameters:

- Number of disks (N) = 5
- Capacity of each disk (C) = 2 TB

Step 2: Apply the RAID-5 capacity formula:

$$\begin{aligned} C_{\text{usable}} &= (5 - 1) \times 2 \text{ TB} \\ &= 4 \times 2 \text{ TB} \\ &= 8 \text{ TB} \end{aligned}$$

Ignoring filesystem formatting overhead, the usable storage capacity available to the user is 8 TB. Options A (6 TB), C (10 TB), and D (12 TB) are incorrect.

Final Answer: 8 TB

Answer: (B)

Go Back to Question 14

Q15.

Solution

Concept: Boolean expressions can be simplified using standard algebraic properties and laws of Boolean algebra:

- **Distributive Law:** $XY + XZ = X(Y + Z)$
- **Complement Law:** $W + \overline{W} = 1$
- **Identity Law:** $W \cdot 1 = W$

Solution: Step 1: Write down the expression to be simplified:

$$F = AB + \overline{A}B + A\overline{B} + \overline{A}\overline{B}$$

Step 2: Group the terms and factor out the common variables: Group the first two terms together and the last two terms together:

$$F = (AB + \overline{A}B) + (A\overline{B} + \overline{A}\overline{B})$$

Step 3: Factor out B from the first group and $\overline{B}$ from the second group:

$$F = B(A + \overline{A}) + \overline{B}(A + \overline{A})$$

Step 4: Apply the complement law, since $A + \overline{A} = 1$:

$$\begin{aligned} F &= B(1) + \overline{B}(1) \\ &= B + \overline{B} \end{aligned}$$

Step 5: Apply the complement law once more:

$$F = B + \overline{B} = 1$$

The simplified expression evaluates to the logical constant 1. This matches Option C.

Final Answer: $\boxed{1}$

Answer: (C)

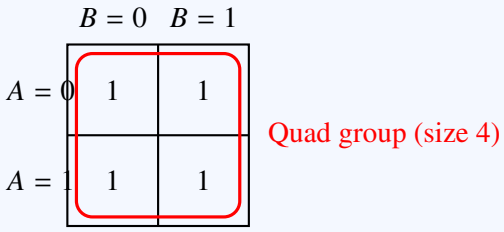
Go Back to Question 15

Q16.

Solution

Concept: In the minimization of Boolean functions using a Karnaugh map (K-map):

- An **implicant** is a grouping of adjacent 1s on a K-map whose size is a power of 2 (2^k).
- A **prime implicant (PI)** is an implicant (group) that cannot be combined with any other adjacent group to form a larger group. In other words, it is a maximal grouping of 1s.



Solution: Step 1: Map the given minterms $m(0, 1, 2, 3)$ onto a 2-variable K-map: A 2-variable K-map contains exactly 4 cells representing the four minterms:

$$m_0 = \overline{A}\,\overline{B}, \quad m_1 = \overline{A}B, \quad m_2 = A\overline{B}, \quad m_3 = AB$$

Since the function is defined as $F(A, B) = \Sigma m(0, 1, 2, 3)$, all four cells in the K-map contain a '1'.

Step 2: Group the 1s into the largest possible adjacent rectangular loops of size 2^k: We can group all four cells together to form a single quad (a group of size $4 = 2^2$).

Step 3: Evaluate if this group can be expanded: The group containing all four cells covers the entire K-map. It cannot be merged with any other cells to form a larger group of size 8, so it represents a prime implicant.

Step 4: Count the total prime implicants: There are no other 1s left on the map, and no other maximal groups can be formed. Thus, there is exactly 1 prime implicant (which simplifies algebraically to the constant value 1).

Option B is correct.

Final Answer: 1

Answer: (B)

Go Back to Question 16

Q17.

Solution

Concept: For a Boolean function of n binary variables:

- There are exactly 2^n unique input combinations (rows in the truth table).
- In the canonical Sum-of-Minterms form, each minterm represents a unique input combination for which the function evaluates to 1.
- The number of input combinations that make the function evaluate to 0 (corresponding to maxterms) is computed by subtracting the number of active minterms from the total possible input combinations:

$$\text{Combinations evaluating to 0} = 2^n - \text{Number of Minterms}$$

Solution: Step 1: Identify the given parameters of the Boolean function:

- Number of variables (n) = 6
- Number of minterms = 20

Step 2: Calculate the total number of input combinations:

$$\begin{aligned}\text{Total combinations} &= 2^n = 2^6 \\ &= 64\end{aligned}$$

Step 3: Calculate the number of input combinations for which the function evaluates to 0:

$$\begin{aligned}\text{Combinations evaluating to 0} &= 64 - 20 \\ &= 44\end{aligned}$$

Thus, exactly 44 input combinations will make the function evaluate to 0. This matches Option C.

Final Answer: 44

Answer: (C)

Go Back to Question 17

Q18.

Solution

Concept: A standard classful Class C IPv4 network has a default subnet mask of 255.255.255.0 (/24), where:

- The first 24 bits are allocated to the network prefix.
- The remaining 8 bits are allocated to the host identifier.

When a Class C network is subnetted using a new subnet mask, some of the 8 host bits are "borrowed" to create subnets. If s bits are borrowed for subnetting, the number of subnets obtained under modern classless routing rules is:

$$\text{Number of Subnets} = 2^s$$

Solution: Step 1: Write down the given subnet mask in binary:

$$255.255.255.240 \rightarrow 11111111.11111111.11111111.11110000_2$$

Step 2: Determine the number of borrowed subnet bits (s): The default mask uses 24 bits. The new mask has 24 + 4 = 28 bits set to 1. Looking at the last octet:

$$240_{10} = 11110000_2$$

This indicates that the first 4 bits of the host portion have been allocated for subnetting:

$$s = 4 \text{ bits}$$

Step 3: Calculate the number of subnets:

$$\begin{aligned}\text{Number of Subnets} &= 2^s \\ &= 2^4 = 16\end{aligned}$$

Thus, exactly 16 subnets are obtained. Option C is correct.

Final Answer: 16

Answer: (C)

Go Back to Question 18

Q19.

Solution

Concept: In relational database schema design, a functional dependency $X \to Y$ is a transitive dependency if there exists a set of attributes Z such that:

(a) $X \to Z$ holds.

(b) $Z \to Y$ holds.

(c) $Z \to X$ does not hold (i.e., Z is not a candidate key or superkey).

Under Armstrong's axioms, when a candidate key X determines Z directly, and Z determines Y, then Y is said to have a transitive dependency on the key X via Z.

Solution: Step 1: Analyze the given relation and dependencies:

- Relation: $R(A, B, C, D, E)$
- Candidate Key: A
- Functional dependencies: $A \to BC$ (meaning $A \to B$ and $A \to C$), $C \to D$, $D \to E$

Step 2: Identify the direct dependencies from the candidate key A:

- $A \to B$ (direct dependency)
- $A \to C$ (direct dependency)

Since B and C are directly determined by the key A, they are not transitively dependent on A.
Step 3: Identify the indirect (transitive) dependencies:

- We have $A \to C$ and $C \to D$. Since C is a non-key attribute ($C \nrightarrow A$), the attribute D is transitively dependent on A through C.
- We also have $A \to D$ (transitively) and $D \to E$. Since D is a non-key attribute ($D \nrightarrow A$), the attribute E is transitively dependent on A through D.

Step 4: Match with the available options: Among the attributes that are transitively dependent on A ($\{D, E\}$), only D is present in the options (Option C). B and C are direct dependencies, and A is the key itself.

Final Answer: D

Answer: (C)

Go Back to Question 19

Q20.

Solution

Concept: The Banker's Algorithm is used by operating systems to safely allocate system resources without entering a deadlock state.

- **Maximum Matrix (Max):** Stores the maximum resource demands of each process.
- **Allocation Matrix (Allocation):** Stores the amount of resources of each type currently allocated to each process.
- **Need Matrix (Need):** Stores the remaining resource demands of each process to successfully finish execution.

For any process i and resource type j, the remaining need is the difference between its maximum declared demand and its currently allocated portion:

$$\text{Need}[i][j] = \text{Max}[i][j] - \text{Allocation}[i][j]$$

Solution: Step 1: Evaluate the matrix arithmetic definitions: The relationship between what a process can request (Need), what it has already received (Allocation), and what it might request at most (Maximum) is defined by:

$$\text{Need} = \text{Maximum} - \text{Allocation}$$

Step 2: Contrast with the options:

- **Option A:** Summing allocation and maximum does not yield the remaining need.
- **Option B:** Correctly defines the subtraction of Allocation from Maximum.
- **Option C:** Yields negative values since Allocation is always less than or equal to Maximum.
- **Option D:** Subtraction from the Available resource vector is not used to define individual process Need vectors.

Thus, Option B is correct.

Final Answer: Need = Maximum - Allocation

Answer: (B)

Go Back to Question 20

Answer Key

Q	Ans	Q	Ans	Q	Ans	Q	Ans	Q	Ans
1	B	2	C	3	D	4	B	5	A
6	A	7	A	8	A	9	C	10	C
11	D	12	B	13	B	14	B	15	C
16	B	17	C	18	C	19	C	20	B