NIMCET Computer Awareness Sample Paper-8

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 48-bit virtual address is translated using a paging system with page size 4 KB.

Ignoring all protection bits and status bits, how many bits are required to represent the virtual page number?

(A) 32

(B) 34

(C) 36

(D) 38

Q2. A processor has a clock frequency of 2.5 GHz and executes a program consisting of 5×10^8 instructions.

If the average CPI is 1.6, what is the execution time of the program?

(A) 0.16 s

(B) 0.24 s

(C) 0.32 s

(D) 0.40 s

Q3. A machine instruction consists of a 6-bit opcode field and three operand fields. The processor contains 64 registers.

What is the minimum instruction length required if each operand must specify a register?

(A) 18 bits

(B) 22 bits

(C) 24 bits

(D) 30 bits

Q4. A direct-mapped cache contains 1024 lines and uses blocks of 16 bytes.

Assuming a 32-bit address space, how many bits are used for the tag field?

(A) 16

(B) 18

(C) 20

(D) 22

Q5. The hexadecimal number

$$(ABCD)_{16}$$

is converted into binary form.

What is the total number of binary 1's present in the result?

(A) 8

(B) 9

(C) 10

(D) 11

Q6. The decimal number 511 is represented in binary form.

How many bits are required in the minimum unsigned binary representation of this number?

(A) 8

(B) 9

(C) 10

(D) 11

Q7. A binary communication channel uses a Hamming code capable of correcting single-bit errors.

If the transmitted codeword contains 11 data bits, what is the minimum number of parity bits required?

(A) 3

(B) 4

(C) 5

(D) 6

Q8. A 12-bit Digital-to-Analog Converter (DAC) is used in an embedded system.

How many distinct analog output levels can be generated by the DAC?

(A) 2048

(B) 4096

(C) 8192

(D) 16384

Q9. In IEEE-754 single precision representation, the exponent field contains the value 126.

Ignoring denormalized numbers, what is the actual exponent represented?

(A) -2

(B) -1

(C) 0

(D) 1

Q10. The Gray code

is converted into binary.

Which of the following binary numbers is obtained?

(A) 110110

(B) 110101

(C) 111001

(D) 111010

Q11. A cache has a hit ratio of 96%. Cache access time is 2 ns and main memory access time is 80 ns.

Assuming serial access, what is the effective memory access time?

(A) 4.8 ns

(B) 5.2 ns

(C) 6.0 ns

(D) 7.0 ns

Q12. A memory chip organization is specified as

$$64K \times 8.$$

What is the storage capacity of the chip?

(A) 32 KB

(B) 64 KB

(C) 128 KB

(D) 256 KB

Q13. A Translation Lookaside Buffer (TLB) is used in virtual memory systems.

What is the primary purpose of the TLB?

(A) Store cache blocks

(B) Store recently used page table entries

(C) Store CPU instructions

(D) Store process control blocks

Q14. A magnetic disk contains 2048 cylinders, 16 heads, and 128 sectors per track. Each sector stores 512 bytes.

What is the total storage capacity of the disk?

(A) 1 GB

(B) 2 GB

(C) 4 GB

(D) 8 GB

Q15. Using Boolean algebra, simplify

$$(A + \overline{B})(A + B) + \overline{A}B.$$

The simplified expression is:

(A) $A + B$

(B) AB

(C) A

(D) B

Q16. The complement of the Boolean function

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

is equal to:

(A) $(\overline{A} + B)(\overline{B} + \overline{C})$

(B) $(A + B)(B + C)$

(C) $(A + \overline{B})(B + C)$

(D) $(\overline{A} + \overline{B})(B + \overline{C})$

Q17. A Boolean function of 8 variables is represented using a Karnaugh map.

How many cells would be required if a complete K-map were constructed?

(A) 64

(B) 128

(C) 256

(D) 512

Q18. A subnet mask of

$$255.255.255.224$$

is used in an IPv4 network.

How many usable host addresses are available in each subnet?

(A) 28

(B) 30

(C) 32

(D) 62

Q19. Consider the relation

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

with functional dependencies

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

Which of the following is a candidate key of the relation?

(A) A

(B) B

(C) BC

(D) AD

Q20. A system allocates resources according to the Banker's Algorithm.

Which of the following best describes a safe state?

(A) No process is waiting

(B) Deadlock has already occurred

(C) There exists at least one safe sequence of process execution

(D) All resources are fully utilized

Detailed Solutions

Q1.

Solution

Concept: In a paging-based virtual memory system, the virtual address generated by the processor is split into two primary fields:

(a) **Virtual Page Number (VPN):** Used to index into page tables for translation.

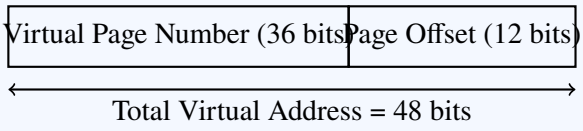
(b) **Page Offset:** Direct byte address index within the physical page.

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

$$\text{Page Offset Bits} = \log_2(S_{\text{page}})$$

Once the offset bits are known, the virtual page number bits can be calculated by subtracting the offset bits from the total virtual address size:

$$\text{Virtual Page Number (VPN) Bits} = \text{Virtual Address Bits} - \text{Page Offset Bits}$$



Solution: Step 1: Convert the page size to bytes:

$$\begin{aligned} S_{\text{page}} &= 4 \text{ KB} \\ &= 4 \times 1024 \text{ Bytes} = 4096 \text{ Bytes} \\ &= 2^2 \times 2^{10} \text{ Bytes} = 2^{12} \text{ Bytes} \end{aligned}$$

Step 2: Determine the number of bits required for the page offset:

$$\text{Page Offset Bits} = \log_2(2^{12}) = 12 \text{ bits}$$

Step 3: Subtract the page offset bits from the total virtual address size (48 bits):

$$\begin{aligned} \text{VPN Bits} &= 48 \text{ bits} - 12 \text{ bits} \\ &= 36 \text{ bits} \end{aligned}$$

Thus, exactly 36 bits are required to represent the virtual page number. Options A (32), B (34), and D (38) are incorrect.

Final Answer: 36

Answer: (C)

Go Back to Question 1

Q2.

Solution

Concept: The execution time (T_{exe}) of a program on a processor can be calculated using the CPU performance equation:

$$T_{\text{exe}} = \text{Instruction Count (IC)} \times \text{Average CPI} \times \text{Clock Cycle Time } (\tau)$$

where the clock cycle time (τ) is the reciprocal of the processor's clock frequency (f):

$$\tau = \frac{1}{f}$$

Substituting τ, the execution time equation becomes:

$$T_{\text{exe}} = \frac{\text{Instruction Count (IC)} \times \text{Average CPI}}{f}$$

Solution: Step 1: Identify the given system parameters:

- Clock frequency (f) = 2.5 GHz = 2.5×10^9 Hz
- Instruction Count (IC) = 5×10^8
- Average CPI = 1.6

Step 2: Calculate the total number of clock cycles required to execute the program:

$$\begin{aligned}\text{Total Clock Cycles} &= IC \times \text{Average CPI} \\ &= (5 \times 10^8) \times 1.6 \\ &= 8 \times 10^8 \text{ cycles}\end{aligned}$$

Step 3: Compute the execution time by dividing the total clock cycles by the clock frequency:

$$\begin{aligned}T_{\text{exe}} &= \frac{8 \times 10^8 \text{ cycles}}{2.5 \times 10^9 \text{ Hz}} \\ &= \frac{8}{25} \text{ s} \\ &= 0.32 \text{ s}\end{aligned}$$

The execution time of the program is 0.32 seconds. Options A (0.16 s), B (0.24 s), and D (0.40 s) are incorrect.

Final Answer: $\boxed{0.32 \text{ s}}$

Answer: (C)

Go Back to Question 2

Q3.

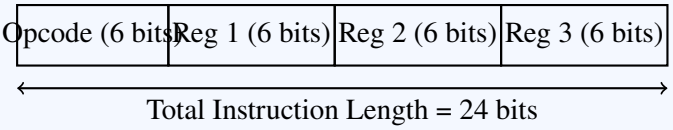
Solution

Concept: An instruction format consists of an opcode field (defining the operation to perform) and operand fields (defining the registers or memory locations containing the data). If a CPU contains R registers, the number of bits required to uniquely select one register is:

$$b = \lceil \log_2(R) \rceil$$

For an instruction template specifying an opcode and k independent register operands, the minimum instruction length (L) is:

$$L = \text{Opcode Bits} + (k \times b)$$



Solution: Step 1: Identify the size of the individual fields:

- Opcode field = 6 bits
- Number of registers (R) = 64
- Number of operands (k) = 3

Step 2: Calculate the number of bits required per register operand:

$$\begin{aligned} \text{Bits per register} &= \log_2(64) \\ &= \log_2(2^6) = 6 \text{ bits} \end{aligned}$$

Step 3: Sum the fields to compute the minimum instruction length:

$$\begin{aligned} \text{Minimum Length} &= \text{Opcode Bits} + (3 \times \text{Bits per register}) \\ &= 6 \text{ bits} + (3 \times 6 \text{ bits}) \\ &= 6 + 18 = 24 \text{ bits} \end{aligned}$$

Thus, the minimum instruction length is 24 bits. Options A (18 bits), B (22 bits), and D (30 bits) are incorrect.

Final Answer: 24 bits

Answer: (C)

Go Back to Question 3

Q4.

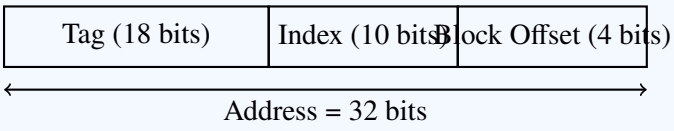
Solution

Concept: In a direct-mapped cache, a memory address is divided into three distinct parts:

$$\text{Memory Address} = \text{Tag} + \text{Index} + \text{Block Offset}$$

where:

- **Block Offset Bits** = $\log_2(\text{Block Size in Bytes})$
- **Index Bits** = $\log_2(\text{Number of Cache Lines})$
- **Tag Bits** = Total Address Space Bits − (Index Bits + Block Offset Bits)



Solution: Step 1: Identify the given memory and cache characteristics:

- Address Space = 32 bits
- Number of cache lines = 1024 = 2^{10}
- Block size = 16 Bytes = 2^4 Bytes

Step 2: Calculate the bits required for the block offset:

$$\text{Block Offset Bits} = \log_2(2^4) = 4 \text{ bits}$$

Step 3: Calculate the bits required for the cache index:

$$\text{Index Bits} = \log_2(1024) = \log_2(2^{10}) = 10 \text{ bits}$$

Step 4: Determine the tag bit width by subtraction:

$$\begin{aligned}\text{Tag Bits} &= 32 - (\text{Index Bits} + \text{Block Offset Bits}) \\ &= 32 - (10 + 4) \\ &= 32 - 14 = 18 \text{ bits}\end{aligned}$$

Hence, 18 bits are used for the tag field. Option B is correct.

Final Answer: 18

Answer: (B)

Go Back to Question 4

Q5.

Solution

Concept: To convert a hexadecimal (base-16) number to binary (base-2), each hexadecimal digit is individually expanded into its equivalent 4-bit binary representation:

$$d_{16} \rightarrow (b_3 b_2 b_1 b_0)_2$$

Once the conversion is complete, the total number of binary 1's can be determined by counting the set bits in each of the converted nibbles.

Solution: Step 1: Map each hexadecimal digit in $(ABCD)_{16}$ to its decimal and 4-bit binary equivalent:

- **A** = $10_{10} = 1010_2$
- **B** = $11_{10} = 1011_2$
- **C** = $12_{10} = 1100_2$
- **D** = $13_{10} = 1101_2$

Step 2: Count the binary 1's present in each digit's representation:

- **A** (1010_2): 2 ones
- **B** (1011_2): 3 ones
- **C** (1100_2): 2 ones
- **D** (1101_2): 3 ones

Step 3: Sum the individual counts:

$$\begin{aligned} \text{Total 1's} &= 2 + 3 + 2 + 3 \\ &= 10 \end{aligned}$$

Step 4: Verify by writing out the complete 16-bit binary sequence:

$$(ABCD)_{16} = 1010\,1011\,1100\,1101_2$$

Counting the 1's in the combined string yields exactly 10. Thus, Option C is correct.

Final Answer: $\boxed{10}$

Answer: (C)

Go Back to Question 5

Q6.

Solution

Concept: To represent a positive decimal integer N using the minimum number of bits b in an unsigned binary representation, we must find the smallest integer b such that:

$$N < 2^b$$

Equivalently, the minimum bit width is:

$$b = \lfloor \log_2(N) \rfloor + 1$$

Solution: Step 1: Identify the decimal value to be converted:

$$N = 511$$

Step 2: Analyze successive powers of 2 near the value:

- $2^8 = 256$
- $2^9 = 512$

Step 3: Establish the upper bound inequality: Since $256 \leq 511 < 512$, we can write:

$$2^8 \leq 511 < 2^9$$

This indicates that 8 bits can only represent values up to 255 ($2^8 - 1$). To represent 511, a minimum of 9 bits is required.

Step 4: Write down the binary representation of 511:

$$\begin{aligned} 511_{10} &= 512 - 1 \\ &= 111111111_2 \end{aligned}$$

This is a sequence of exactly nine 1's. Thus, 9 bits are required. Option B is correct.

Final Answer: $\boxed{9}$

Answer: (B)

Go Back to Question 6

Q7.

Solution

Concept: In single-bit error-correcting Hamming codes, the relationship between the number of data bits (d) and the number of parity bits (p) must satisfy the Hamming rule inequality:

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

The p parity bits generate 2^p unique syndrome values, which must cover all possible single-bit error positions (including d data bits and p parity bits) plus a "no error" status.

Solution: Step 1: Set up the inequality using the given number of data bits, $d = 11$:

$$2^p \geq 11 + p + 1$$
$$2^p \geq p + 12$$

Step 2: Test successive values of p starting from 3 to find the smallest valid integer:

- **Test $p = 3$:**

$$2^3 \geq 3 + 12 \implies 8 \geq 15 \quad \text{(False)}$$

- **Test $p = 4$:**

$$2^4 \geq 4 + 12 \implies 16 \geq 16 \quad \text{(True)}$$

Since $p = 4$ is the smallest integer that satisfies the inequality, a minimum of 4 parity bits are required. Options A (3), C (5), and D (6) are incorrect.

Final Answer: $\boxed{4}$

Answer: (B)

Go Back to Question 7

Q8.

Solution

Concept: A Digital-to-Analog Converter (DAC) takes an n-bit binary input and converts it into a proportional analog output level (voltage or current). The number of distinct analog output levels that a DAC can produce is directly dictated by the number of unique binary input combinations possible with n bits:

$$\text{Distinct Output Levels} = 2^n$$

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

$$n = 12 \text{ bits}$$

Step 2: Calculate the number of distinct binary input codes:

$$\text{Distinct levels} = 2^{12}$$

Step 3: Evaluate the exponent:

$$2^{12} = 4096$$

Thus, a 12-bit DAC can generate 4096 distinct analog output levels (typically representing voltages from 0 to $\frac{4095}{4096}V_{\text{ref}}$).
Options A (2048), C (8192), and D (16384) are incorrect. Option B is correct.

Final Answer: 4096

Answer: (B)

Go Back to Question 8

Q9.

Solution

Concept: Under the IEEE-754 single-precision representation standard, the exponent field is stored as an 8-bit biased integer. To represent both fractional and large numbers, a bias of 127 is added to the actual exponent (E) to compute the stored biased exponent (E_{biased}):

$$E_{\text{biased}} = E + \text{Bias}$$

To recover the actual exponent from the stored biased value, the relationship is:

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

where Bias = 127 for single precision.

Solution: Step 1: Identify the given biased exponent value from the exponent field:

$$E_{\text{biased}} = 126$$

Step 2: Subtract the single-precision bias of 127 to determine the actual exponent:

$$\begin{aligned} E &= 126 - 127 \\ &= -1 \end{aligned}$$

Step 3: Verify the bounds: Since $E_{\text{biased}} = 126$ falls in the normalized exponent range $[1, 254]$ (where $E_{\text{biased}} = 0$ is reserved for denormals/zero and $E_{\text{biased}} = 255$ is reserved for infinity/NaN), the calculation is valid.
The actual exponent represented is -1. Options A (-2), C (0), and D (1) are incorrect.

Final Answer: -1

Answer: (B)

Go Back to Question 9

Q10.

Solution

Concept: To convert Gray code G to binary B:

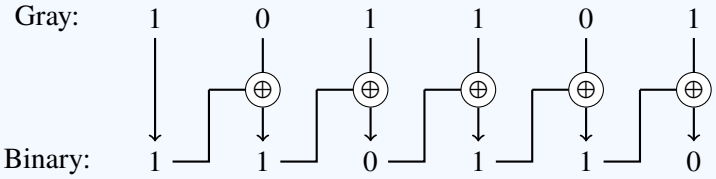
- The MSB remains unchanged:

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

- Each remaining binary bit is obtained using XOR:

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

Thus, binary bits are generated from left to right using successive XOR operations.



Solution: Step 1: Identify the given 6-bit Gray code:

$$G = 101101_2 \quad (g_5 = 1,\ g_4 = 0,\ g_3 = 1,\ g_2 = 1,\ g_1 = 0,\ g_0 = 1)$$

Step 2: Apply the conversion process bit-by-bit from MSB to LSB:

- $b_5 = g_5 = 1$
- $b_4 = b_5 \oplus g_4 = 1 \oplus 0 = 1$
- $b_3 = b_4 \oplus g_3 = 1 \oplus 1 = 0$
- $b_2 = b_3 \oplus g_2 = 0 \oplus 1 = 1$
- $b_1 = b_2 \oplus g_1 = 1 \oplus 0 = 1$
- $b_0 = b_1 \oplus g_0 = 1 \oplus 1 = 0$

Step 3: Combine the calculated binary bits:

$$B = 110110_2$$

Comparing with the options, this matches Option A. Options B (110101), C (111001), and D (111010) do not represent the correct binary conversion.

Final Answer: 110110

Answer: (A)

Go Back to Question 10

Q11.

Solution

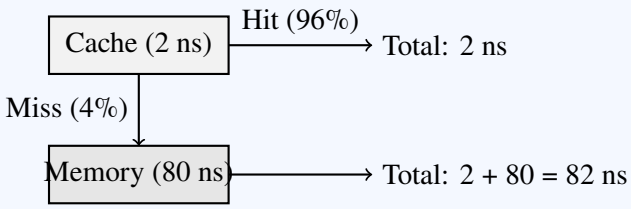
Concept: Under a serial (sequential) memory access scheme, the cache memory is always accessed first.

- If there is a **cache hit**, the data is successfully retrieved directly from the cache, taking T_{cache} time.
- If there is a **cache miss**, the cache access still takes T_{cache} time, after which the main memory must be accessed, taking an additional T_{mem} time.

Thus, the effective memory access time (T_{eff}) is computed as:

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

where h is the hit ratio, T_{cache} is the cache access time, and T_{mem} is the main memory access time.



Solution: Step 1: Identify the given system parameters:

- Hit ratio (h) = 96% = 0.96
- Miss ratio ($1-h$) = 4% = 0.04
- Cache access time (T_{cache}) = 2 ns
- Main memory access time (T_{mem}) = 80 ns

Step 2: Apply the parameters to the serial access formula:

$$\begin{aligned} T_{\text{eff}} &= T_{\text{cache}} + (1-h) \cdot T_{\text{mem}} \\ &= 2 \text{ ns} + (0.04 \times 80 \text{ ns}) \\ &= 2 \text{ ns} + 3.2 \text{ ns} \\ &= 5.2 \text{ ns} \end{aligned}$$

Thus, the effective memory access time is 5.2 ns. Options A (4.8 ns), C (6.0 ns), and D (7.0 ns) are incorrect.

Final Answer: 5.2 ns

Answer: (B)

Go Back to Question 11

Q12.

Solution

Concept: The capacity of a memory chip specified in the format $W \times B$ is computed by multiplying the number of addressable storage units (W) by the bit width of each storage unit (B):

$$\text{Total Capacity (Bits)} = W \times B$$

To convert the total capacity into standard byte-oriented units (Kilobytes, KB), we use the following definitions:

- 1 K = 1024 addressable units.
- 1 Byte = 8 bits.
- 1 KB = 1024 Bytes.

Solution: Step 1: Identify the specification elements of the 64K $\times$ 8 chip:

- Number of addressable units (W) = 64K = 64×1024
- Bit width per unit (B) = 8 bits

Step 2: Calculate the capacity in bytes: Since 8 bits = 1 Byte, each of the 64K locations stores exactly 1 Byte of information:

$$\begin{aligned} \text{Total Capacity} &= 64\text{K} \times 8 \text{ bits} \\ &= 64\text{K} \times 1 \text{ Byte} \\ &= 64 \text{ KB} \end{aligned}$$

Thus, the total storage capacity of the chip is 64 KB, matching Option B. Options A (32 KB), C (128 KB), and D (256 KB) are incorrect.

Final Answer: 64 KB

Answer: (B)

Go Back to Question 12

Q13.

Solution

Concept: In virtual memory systems, translating a virtual address into a physical address requires mapping through a page table stored in the physical main memory. Because querying main memory for every instruction fetch or operand access creates a significant performance bottleneck, processors use a specialized hardware cache.

This cache is called the **Translation Lookaside Buffer (TLB)**. The TLB is a fast, associative memory that stores a subset of recently accessed page table entries (virtual page numbers mapped to physical page frames). When a translation request is made, the MMU checks the TLB first (TLB hit) before falling back to physical memory (TLB miss).

Solution: Step 1: Evaluate each of the provided options:

- **Option A (Store cache blocks):** Cache blocks are stored in the L1/L2/L3 data and instruction caches, not the TLB.
- **Option B (Store recently used page table entries):** This matches the hardware definition and design purpose of the TLB.
- **Option C (Store CPU instructions):** CPU instructions are stored in the instruction cache (I-Cache) and main memory.
- **Option D (Store process control blocks):** Process Control Blocks (PCBs) are operating system data structures maintained in main memory.

Thus, Option B is correct.

Final Answer: B

Answer: (B)

Go Back to Question 13

Q14.

Solution

Concept: The storage capacity of a magnetic disk using Cylinder-Head-Sector (CHS) geometry is determined by multiplying all physical bounds together:

$$\text{Total Capacity} = \text{Cylinders} \times \text{Heads} \times \text{Sectors per Track} \times \text{Bytes per Sector}$$

Converting the output value to Gigabytes (GB) relies on the binary scale where:

$$1 \text{ GB} = 2^{30} \text{ Bytes} = 1,073,741,824 \text{ Bytes}$$

Solution: Step 1: Identify the given physical parameters of the magnetic disk:

- Cylinders = $2048 = 2^{11}$
- Heads = $16 = 2^4$
- Sectors per Track = $128 = 2^7$
- Bytes per Sector = $512 = 2^9$

Step 2: Compute the total capacity in bytes using power-of-two exponent addition:

$$\begin{aligned}\text{Total Capacity} &= 2^{11} \times 2^4 \times 2^7 \times 2^9 \text{ Bytes} \\ &= 2^{11+4+7+9} \text{ Bytes} \\ &= 2^{31} \text{ Bytes}\end{aligned}$$

Step 3: Convert the byte count to Gigabytes:

$$\begin{aligned}2^{31} \text{ Bytes} &= 2^1 \times 2^{30} \text{ Bytes} \\ &= 2 \times 1 \text{ GB} = 2 \text{ GB}\end{aligned}$$

The total storage capacity of the disk is 2 GB. Options A (1 GB), C (4 GB), and D (8 GB) are incorrect.

Final Answer: $\boxed{2 \text{ GB}}$

Answer: (B)

Go Back to Question 14

Q15.

Solution

Concept: To simplify Boolean expressions, we can apply the foundational rules of Boolean algebra:

- **Distributive Law:** $(X + Y)(X + Z) = X + YZ$
- **Complement Law:** $W \cdot \overline{W} = 0$ and $W + \overline{W} = 1$
- **Identity Law:** $W + 0 = W$
- **Distributive Law of Absorption:** $X + \overline{X}Y = (X + \overline{X})(X + Y) = 1 \cdot (X + Y) = X + Y$

Solution: Step 1: Write down the expression to simplify:

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

Step 2: Simplify the product portion $(A + \overline{B})(A + B)$: Using the distributive law of addition over multiplication, we can factor out A:

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

Since $\overline{B}B = 0$ by the complement law:

$$A + \overline{B}B = A + 0 = A$$

Step 3: Substitute the simplified value back into the original expression:

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

Step 4: Distribute A over the terms using the rule $X + \overline{X}Y = X + Y$:

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

Since $A + \overline{A} = 1$:

$$F = 1 \cdot (A + B) = A + B$$

The simplified expression is $A + B$, which corresponds to Option A.

Final Answer: A+B

Answer: (A)

Go Back to Question 15

Q16.

Solution

Concept: The complement of a Boolean function F (denoted $\overline{F}$) is found by applying De Morgan's Laws:

- **Law 1:** $\overline{X+Y} = \overline{X} \cdot \overline{Y}$
- **Law 2:** $\overline{X \cdot Y} = \overline{X} + \overline{Y}$

By systematically applying these theorems and double negation ($\overline{\overline{W}} = W$), we can resolve the negated form of any complex Boolean expression.

Solution: Step 1: Write down the negation of the function $F = A\overline{B} + BC$:

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

Step 2: Apply the first De Morgan Law to split the principal sum (OR):

$$\overline{F} = (\overline{A\overline{B}}) \cdot (\overline{BC})$$

Step 3: Apply the second De Morgan Law to each independent product (AND) term:

$$\overline{A\overline{B}} = \overline{A} + \overline{\overline{B}} = \overline{A} + B$$
$$\overline{BC} = \overline{B} + \overline{C}$$

Step 4: Substitute these parts back into the product:

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

This is the exact form shown in Option A. Options B, C, and D are incorrect.

Final Answer: $\boxed{(\overline{A} + B)(\overline{B} + \overline{C})}$

Answer: (A)

Go Back to Question 16

Q17.

Solution

Concept: A Karnaugh map (K-map) is a graphical representation used to simplify Boolean expressions.

- Each cell in a K-map corresponds to a single, unique minterm (or input combination) of the truth table.
- For a Boolean function containing n variables, the total number of distinct input combinations in its domain is 2^n.
- Therefore, a complete K-map constructed for an n-variable function requires exactly 2^n cells.

Solution: Step 1: Identify the number of independent variables:

$$n = 8$$

Step 2: Calculate the required number of cells:

$$\begin{aligned} \text{Number of Cells} &= 2^n \\ &= 2^8 \end{aligned}$$

Step 3: Compute the exponent value:

$$2^8 = 256$$

Thus, a complete K-map of 8 variables contains 256 cells. Options A (64), B (128), and D (512) are incorrect.

Final Answer: $\boxed{256}$

Answer: (C)

Go Back to Question 17

Q18.

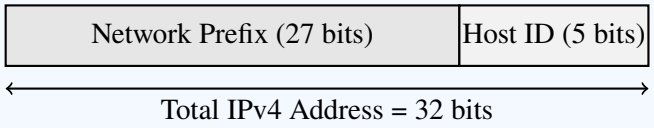
Solution

Concept: An IPv4 address is 32 bits wide, divided into a network portion and a host portion by the subnet mask.

- The bits set to `1` in the mask designate the network/subnet prefix.
- The bits set to `0` in the mask designate the host identifier.

If there are h bits allocated to the host portion, the total number of addresses in the subnet is 2^h. However, because the first address (subnet identifier) and the last address (subnet broadcast address) are reserved and cannot be assigned to individual devices, the number of **usable host addresses** is:

$$\text{Usable Hosts} = 2^h - 2$$



Solution: Step 1: Convert the subnet mask 255.255.255.224 into binary:

$$255.255.255.224 \rightarrow \underbrace{11111111}_{8} \cdot \underbrace{11111111}_{8} \cdot \underbrace{11111111}_{8} \cdot \underbrace{11100000}_{8}$$

Step 2: Determine the number of host bits (h): The last octet contains exactly five `0` bits. Since all other octets consist entirely of `1`s, we have:

$$h = 5 \text{ bits}$$

Step 3: Calculate the total and usable host addresses:

$$\begin{aligned} \text{Total Addresses} &= 2^5 = 32 \\ \text{Usable Host Addresses} &= 2^5 - 2 \\ &= 32 - 2 = 30 \end{aligned}$$

Thus, each subnet has 30 usable host addresses. Option B is correct.

Final Answer: 30

Answer: (B)

Go Back to Question 18

Q19.

Solution

Concept: A set of attributes K is a candidate key for a relation if:

- K^+ contains all attributes of the relation.
- No proper subset of K has this property.

The closure K^+ is found by repeatedly applying the given functional dependencies.

Solution: Step 1: Given relation and functional dependencies:

- Relation: $R(A, B, C, D)$
- FDs: $A \to B$, $A \to C$, $BC \to D$

Step 2: Compute the closure of A:

$$A^+ = \{A\}$$

Using $A \to B$ and $A \to C$:

$$A^+ = \{A, B, C\}$$

Since both B and C are present, apply $BC \to D$:

$$A^+ = \{A, B, C, D\}$$

Step 3: Since A^+ contains all attributes of R, A is a superkey.

Step 4: As A is a single attribute, it has no proper subset. Therefore, it is minimal and hence a candidate key.

Final Answer: A

Answer: (A)

Go Back to Question 19

Q20.

Solution

Concept: The Banker's Algorithm is a deadlock avoidance algorithm that dynamically determines whether allocating resources to a process will keep the system in a safe state.

A state is declared safe if there exists at least one order (referred to as a safe sequence) in which all active processes can complete execution. Specifically, for a sequence of processes $\langle P_1, P_2, \ldots, P_n \rangle$ to be safe, the maximum resources that each process P_i can still request can be satisfied by the currently available resources plus the resources currently held by all preceding processes P_j (where $j < i$). A safe state guarantees that the system can avoid deadlocks.

Solution: Step 1: Evaluate the definition of a safe state: A safe state is not one where no process is waiting (waiting can occur in any safe scheduling state), nor does it mean all resources are fully utilized. Instead, it is defined entirely by the existence of a path (a safe sequence) that guarantees deadlock avoidance.

Step 2: Match with the options:

- **Option A (No process is waiting):** Processes can wait even in a safe state as long as their eventual execution is guaranteed.
- **Option B (Deadlock has already occurred):** A safe state is by definition a state where deadlock cannot occur.
- **Option C (There exists at least one safe sequence of process execution):** This is the exact formal definition of a safe state.
- **Option D (All resources are fully utilized):** Full utilization is not a requirement of a safe state.

Thus, Option C is correct.

Final Answer: C

Answer: (C)

Go Back to Question 20

Answer Key

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