NIMCET Computer Awareness Sample Paper-7

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 cache memory of size 128 KB is 4-way set associative. The block size is 32 bytes and the processor uses 36-bit physical addresses.

How many bits are required for the set index field of the address?

(A) 8

(B) 10

(C) 12

(D) 14

Q2. A processor executes 1 million instructions. The instruction mix is as follows:

40% arithmetic instructions requiring 1 cycle each, 35% memory instructions requiring 2 cycles each, and 25% branch instructions requiring 3 cycles each.

What is the average CPI (Cycles Per Instruction) of the processor?

(A) 1.65

(B) 1.75

(C) 1.85

(D) 1.95

Q3. A CPU has 16 general-purpose registers. Each instruction contains fields for an opcode and three register operands.

What is the minimum number of bits required to encode the three register operands?

(A) 8

(B) 10

(C) 12

(D) 16

Q4. A processor uses little-endian byte ordering.

The 32-bit hexadecimal value

$$12345678_{16}$$

is stored starting at memory address 1000.

Which byte is stored at address 1000?

(A) 12

(B) 34

(C) 56

(D) 78

Q5. The decimal number

$$625$$

is represented in hexadecimal notation.

Which of the following is the correct hexadecimal equivalent?

(A) 261

(B) 271

(C) 281

(D) 291

Q6. The binary number

11101101

is interpreted as an unsigned integer.

What is its decimal equivalent?

(A) 235

(B) 237

(C) 239

(D) 241

Q7. A Hamming code is used for single-bit error correction.

If 4 data bits are to be transmitted, what is the minimum number of parity bits required?

(A) 2

(B) 3

(C) 4

(D) 5

Q8. A 10-bit Analog-to-Digital Converter (ADC) is used to digitize a voltage signal.

How many distinct quantization levels can the ADC produce?

(A) 512

(B) 1000

(C) 1024

(D) 2048

Q9. In IEEE-754 single precision format, the exponent field contains the binary value

10000001

What is the actual exponent represented by this field?

(A) 1

(B) 2

(C) 127

(D) 129

Q10. A binary number contains exactly 12 bits.

How many different unsigned integers can be represented excluding zero?

(A) 4094

(B) 4095

(C) 4096

(D) 8191

Q11. A virtual memory system uses pages of size 8 KB.

If the logical address space contains 30 bits, how many bits are used for the page offset?

(A) 10

(B) 11

(C) 12

(D) 13

Q12. A memory module is organized as

$$2^{20} \times 16$$

What is the total storage capacity of the module?

(A) 1 MB

(B) 2 MB

(C) 4 MB

(D) 16 MB

Q13. In a paging system, a page fault occurs whenever:

Which of the following situations takes place?

(A) A page is modified

(B) A page is accessed for the first time

(C) The required page is not present in main memory

(D) Cache memory becomes full

Q14. Consider a disk having 4096 cylinders, 32 heads and 128 sectors per track. Each sector stores 512 bytes.

Approximately how much data can be stored on the disk?

(A) 4 GB

(B) 8 GB

(C) 16 GB

(D) 32 GB

Q15. Using Boolean algebra, simplify the expression

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

The simplified result is:

(A) AB

(B) $A + B$

(C) A

(D) B

Q16. The dual of the Boolean expression

$$A + BC$$

is:

(A) $A(B + C)$

(B) $AB + C$

(C) $(A + B)C$

(D) $AB + AC$

Q17. A Boolean function of 7 variables is represented in canonical Sum-of-Minterms form.

How many possible minterms exist for this function?

(A) 64

(B) 128

(C) 256

(D) 512

Q18. The IPv6 protocol uses addresses of length 128 bits.

How many times larger is the IPv6 address space compared to IPv4?

(A) 2^{32}

(B) 2^{64}

(C) 2^{96}

(D) 2^{128}

Q19. A relation contains the attributes

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

and the functional dependencies

$$A \rightarrow B, \quad B \rightarrow C, \quad C \rightarrow D$$

If A is a candidate key, which dependency demonstrates a transitive dependency?

(A) $A \rightarrow B$

(B) $B \rightarrow C$

(C) $A \rightarrow D$

(D) $C \rightarrow D$

Q20. A system contains 5 processes and 3 resource types. A deadlock avoidance algorithm decides whether a resource request can be safely granted.

Which operating-system algorithm is specifically designed for this purpose?

(A) FIFO Algorithm

(B) Round Robin Algorithm

(C) Banker's Algorithm

(D) LRU Algorithm

Detailed Solutions

Q1.

Solution

Concept: In a set-associative cache, the physical address is divided into:

- **Block Offset:** $\log_2(\text{Block Size})$ bits
- **Set Index:** $\log_2(\text{Number of Sets})$ bits
- **Tag:** Remaining address bits

The number of sets is:

$$\text{Number of Sets} = \frac{\text{Cache Size}}{\text{Block Size} \times \text{Associativity}}$$

Hence, Set Index bits = $\log_2(\text{Number of Sets})$.

Solution: Step 1: Convert the given parameters:

- Cache Size = 128 KB = 2^{17} Bytes
- Block Size = 32 Bytes = 2^5 Bytes
- Associativity = 4 = 2^2

Step 2: Calculate the number of sets:

$$\begin{aligned}\text{Number of Sets} &= \frac{\text{Cache Size}}{\text{Block Size} \times \text{Associativity}} \\ &= \frac{2^{17}}{2^5 \times 2^2} \\ &= 2^{10}\end{aligned}$$

Step 3: Find the set index bits:

$$\begin{aligned}\text{Set Index Bits} &= \log_2(2^{10}) \\ &= 10\end{aligned}$$

Final Answer: $\boxed{10}$

Answer: (B)

Go Back to Question 1

Q2.

Solution

Concept: The average Cycles Per Instruction (CPI) of a processor measures the average number of clock cycles spent to execute an instruction in a given workload. When an instruction mix is specified along with individual cycle requirements, the overall average CPI is computed as the weighted average of the instruction CPIs:

$$\text{Average CPI} = \sum_{i=1}^{k} (\text{Fraction of instruction type } i \times \text{CPI of instruction type } i)$$

Arithmetic (40%)	Memory (35%)	Branch (25%)
1 Cycle	2 Cycles	3 Cycles

Solution:

Step 1: Note the instruction mix and their cycle requirements:

- Arithmetic: 40% of instructions, 1 cycle each
- Memory: 35% of instructions, 2 cycles each
- Branch: 25% of instructions, 3 cycles each

Step 2: Calculate the weighted average CPI:

$$\begin{aligned} \text{CPI} &= (0.40 \times 1) + (0.35 \times 2) + (0.25 \times 3) \\ &= 0.40 + 0.70 + 0.75 \\ &= 1.85 \end{aligned}$$

Thus, the processor executes each instruction in an average of 1.85 clock cycles. This confirms that Option C is correct, and Options A (1.65), B (1.75), and D (1.95) are incorrect.

Final Answer: 1.85

Answer: (C)

Go Back to Question 2

Q3.

Solution

Concept: In processor instruction set design, register specifier fields are used to identify the registers involved in an operation. If a CPU has N registers, representing any single register uniquely requires a binary code of length b bits, where:

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

For an instruction that contains k independent register operand fields, the total bit budget required to represent all register specifiers is:

$$\begin{aligned} \text{Total Operand Bits} &= k \times b \\ &= k \times \lceil \log_2(N) \rceil \end{aligned}$$

Solution: Step 1: Find the number of bits needed to address a single register: The CPU contains 16 general-purpose registers. To distinguish each register uniquely:

$$\begin{aligned} \text{Bits per register} &= \log_2(16) \\ &= \log_2(2^4) = 4 \text{ bits} \end{aligned}$$

Thus, each of the 16 registers can be mapped to a unique binary sequence from `0000` to `1111`. Step 2: Determine the bits required for three independent register operands: Since each of the three operands (such as two source registers and one destination register) must be specified independently, each requires its own 4-bit field:

$$\begin{aligned} \text{Total Bits} &= 3 \times \text{Bits per register} \\ &= 3 \times 4 \text{ bits} = 12 \text{ bits} \end{aligned}$$

Thus, a minimum of 12 bits is required. Options A (8), B (10), and D (16) do not represent the correct minimum limit. Hence, Option C is correct.

Final Answer: 12

Answer: (C)

Go Back to Question 3

Q4.

Solution

Concept: Byte-ordering methods (endianness) dictate how a multi-byte word is mapped to sequential memory addresses:

- **Little-Endian:** The least significant byte (LSB), representing the lowest-order numerical value, is stored at the lowest (starting) physical memory address. Higher-order bytes are stored at sequentially higher addresses.
- **Big-Endian:** The most significant byte (MSB), representing the highest-order numerical value, is stored at the lowest (starting) physical memory address.

Address 1003:	12 (MSB / Byte 3)
Address 1002:	34 (Byte 2)
Address 1001:	56 (Byte 1)
Address 1000:	78 (LSB / Byte 0)

Solution: Step 1: Deconstruct the 32-bit hexadecimal number 12345678_{16} into 4 byte-sized components (where each byte consists of two hexadecimal digits):

- **Byte 3 (MSB):** 12_{16}
- **Byte 2:** 34_{16}
- **Byte 1:** 56_{16}
- **Byte 0 (LSB):** 78_{16}

Step 2: Map these bytes to memory locations starting at address 1000 using little-endian byte ordering:

$$\text{Address } 1000 \rightarrow \text{Byte } 0 \text{ (LSB)} = 78_{16}$$
$$\text{Address } 1001 \rightarrow \text{Byte } 1 = 56_{16}$$
$$\text{Address } 1002 \rightarrow \text{Byte } 2 = 34_{16}$$
$$\text{Address } 1003 \rightarrow \text{Byte } 3 \text{ (MSB)} = 12_{16}$$

Step 3: Analyze the question and identify the byte at the starting address (1000): The byte stored at address 1000 is 78 (Option D).
(Note: In a big-endian system, the byte at address 1000 would instead be 12, which corresponds to Option A).

Final Answer: 78

Answer: (D)

Go Back to Question 4

Q5.

Solution

Concept: Converting a decimal value (base-10) to a hexadecimal value (base-16) requires finding the coefficients of powers of 16 that sum to the original number:

$$N_{10} = a_k \cdot 16^k + a_{k-1} \cdot 16^{k-1} + \cdots + a_1 \cdot 16^1 + a_0 \cdot 16^0$$

The standard conversion algorithm relies on dividing the decimal integer by 16 repeatedly. The successive remainders produced during division correspond to the digits $a_0, a_1, \ldots$ of the hexadecimal representation, read from the last non-zero remainder to the first.

Solution: Step 1: Divide the decimal number 625 by 16 and record the quotient and remainder:

$$625 \div 16 = 39 \quad \text{with a remainder of } 1 \quad \rightarrow (a_0 = 1)$$

Since the quotient 39 is greater than or equal to 16, we continue dividing.
Step 2: Divide the quotient 39 by 16:

$$39 \div 16 = 2 \quad \text{with a remainder of } 7 \quad \rightarrow (a_1 = 7)$$

Step 3: Divide the quotient 2 by 16:

$$2 \div 16 = 0 \quad \text{with a remainder of } 2 \quad \rightarrow (a_2 = 2)$$

Since the quotient is now 0, the division loop terminates.
Step 4: Collect the remainders from the last division step up to the first ($a_2 a_1 a_0$):

$$(625)_{10} = (271)_{16}$$

Step 5: Verify the hexadecimal representation by converting back to decimal:

$$\begin{aligned}(271)_{16} &= 2 \cdot 16^2 + 7 \cdot 16^1 + 1 \cdot 16^0 \\ &= 2 \cdot 256 + 7 \cdot 16 + 1 \\ &= 512 + 112 + 1 = 625\end{aligned}$$

This verifies that 271 is indeed the correct representation. Options A (261), C (281), and D (291) are incorrect.

Final Answer: 271

Answer: (B)

Go Back to Question 5

Q6.

Solution

Concept: An n-bit unsigned binary number is written as $(b_{n-1}b_{n-2}\ldots b_0)_2$. Its base-10 decimal equivalent is obtained by calculating the sum of each bit multiplied by its respective power-of-two weight:

$$\text{Decimal Value} = b_{n-1}\cdot 2^{n-1} + b_{n-2}\cdot 2^{n-2} + \cdots + b_1\cdot 2^1 + b_0\cdot 2^0$$

Solution: Step 1: Write down the given binary number 11101101_2 and align each bit with its positional weight:

Bit Position (i) :	7	6	5	4	3	2	1	0
Bit Value (b_i) :	1	1	1	0	1	1	0	1
Power of 2 (2^i) :	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Weight (Decimal):	128	64	32	16	8	4	2	1

Step 2: Add the decimal weights of only the active positions where the bit is 1:

$$\begin{aligned}\text{Decimal Value} &= (1\cdot 128) + (1\cdot 64) + (1\cdot 32) + (0\cdot 16) + (1\cdot 8) + (1\cdot 4) + (0\cdot 2) + (1\cdot 1)\\ &= 128 + 64 + 32 + 8 + 4 + 1\end{aligned}$$

Step 3: Perform the additions:

$$\begin{aligned}128 + 64 + 32 &= 224\\ 8 + 4 + 1 &= 13\\ \text{Total} &= 224 + 13 = 237\end{aligned}$$

Step 4: Alternative check using subtraction from the maximum possible 8-bit value (255): The maximum value of an 8-bit unsigned integer is $11111111_2 = 255$. The inactive bits (0s) in 11101101_2 are at positions 4 ($2^4 = 16$) and 1 ($2^1 = 2$).

$$\text{Decimal Value} = 255 - 16 - 2 = 237$$

Both calculation methods yield 237. Options A (235), C (239), and D (241) are incorrect.

Final Answer: 237

Answer: (B)

Go Back to Question 6

Q7.

Solution

Concept: Hamming codes are structured for single-bit error correction (SEC). To correct an error, the code must generate a unique syndrome for each potential error location, as well as a syndrome representing the "no-error" condition.

If we have d data bits and p parity (redundancy) bits, the total length of the codeword is $d + p$. Each of these $d + p$ positions can experience a single-bit error. Therefore, the p parity bits must provide at least $d + p + 1$ unique binary combinations:

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

Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
p_1 (Parity)	p_2 (Parity)	d_1 (Data)	p_3 (Parity)	d_2 (Data)	d_3 (Data)	d_4 (Data)

Solution: Step 1: Set up the inequality with the given number of data bits, $d = 4$:

$$2^p \geq 4 + p + 1$$
$$2^p \geq p + 5$$

Step 2: Test successive values of p to find the smallest integer that satisfies this relation:

- **Test $p = 1$:**

$$2^1 \geq 1 + 5 \implies 2 \geq 6 \quad \text{(False)}$$

- **Test $p = 2$:**

$$2^2 \geq 2 + 5 \implies 4 \geq 7 \quad \text{(False)}$$

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

$$2^3 \geq 3 + 5 \implies 8 \geq 8 \quad \text{(True)}$$

Since $p = 3$ is the first integer value of p that satisfies the inequality, a minimum of 3 parity bits are required. The resulting code is a $(7, 4)$ Hamming code, where the codeword length is 7 bits. Options A (2), C (4), and D (5) are incorrect.

Final Answer: $\boxed{3}$

Answer: (B)

Go Back to Question 7

Q8.

Solution

Concept: An Analog-to-Digital Converter (ADC) translates a continuous analog voltage signal into discrete digital values. The precision of this conversion is dictated by the resolution of the ADC, expressed in bits (n). An n-bit ADC maps the continuous input range into a finite number of discrete levels, which is given by:

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

Each increment in the resolution by 1 bit doubles the number of available quantization levels.
Solution: Step 1: Identify the resolution of the ADC from the problem statement:

$$n = 10 \text{ bits}$$

Step 2: Calculate the number of distinct digital values (quantization levels) that can be generated:

$$\begin{aligned}\text{Distinct levels} &= 2^n \\ &= 2^{10}\end{aligned}$$

Step 3: Evaluate the power of two:

$$2^{10} = 1024$$

A 10-bit converter can resolve its input into exactly 1024 separate steps. This means that if the analog range is 0V to V_{ref}, each step (or LSB size) is $\frac{V_{\text{ref}}}{1024}$.

Options A (512, which corresponds to 2^9), B (1000, which is a round decimal value but not a power of 2), and D (2048, which corresponds to 2^{11}) are incorrect.

Final Answer: $\boxed{1024}$

Answer: (C)

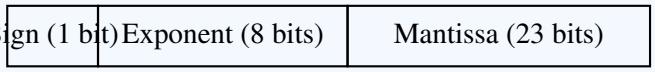
Go Back to Question 8

Q9.

Solution

Concept: The IEEE-754 standard for single-precision floating-point representation allocates 32 bits as follows:

- **Sign bit (S):** 1 bit (bit 31)
- **Biased Exponent (E_{biased}):** 8 bits (bits 23 to 30)
- **Fraction / Mantissa (M):** 23 bits (bits 0 to 22)



To allow for both positive and negative exponents without needing an explicit sign bit inside the exponent field, a bias is added. The actual exponent (E) is recovered using the formula:

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

For single-precision, the bias is 127.

Solution: Step 1: Convert the binary exponent field value (10000001_2) to decimal: The 8-bit binary pattern is evaluated as:

$$\begin{aligned} E_{\text{biased}} &= (1 \cdot 2^7) + (0 \cdot 2^6) + (0 \cdot 2^5) + (0 \cdot 2^4) + (0 \cdot 2^3) + (0 \cdot 2^2) + (0 \cdot 2^1) + (1 \cdot 2^0) \\ &= 128 + 0 + 0 + 0 + 0 + 0 + 0 + 1 \\ &= 129 \end{aligned}$$

Step 2: Apply the subtraction of the standard single-precision bias (127) to find the actual exponent:

$$\begin{aligned} E &= E_{\text{biased}} - \text{Bias} \\ &= 129 - 127 \\ &= 2 \end{aligned}$$

Thus, the actual exponent represented by this field is 2. Options A (1), C (127), and D (129) are incorrect.

Final Answer: $\boxed{2}$

Answer: (B)

Go Back to Question 9

Q10.

Solution

Concept: An n-bit binary number has exactly 2^n unique bit combinations (or states). In unsigned integer representation, these combinations map to integers starting from 0 up to $2^n - 1$:

$$\text{Unsigned Range} = [0, 2^n - 1]$$

The total count of values in this range is 2^n. If we are asked to find the number of representable unsigned integers while explicitly excluding zero, we omit the value representing zero. The count of non-zero unsigned integers is:

$$\text{Non-Zero Count} = 2^n - 1$$

Solution: Step 1: Identify the number of bits:

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

Step 2: Calculate the total number of unique binary configurations:

$$2^{12} = 4096$$

These 4096 states represent the set of integers $\{0, 1, 2, 3, \ldots, 4095\}$.
Step 3: Subtract the single state representing zero: To find the number of unsigned integers excluding zero, we subtract 1 (the code representing 0) from the total:

$$\begin{aligned} \text{Count} &= 2^{12} - 1 \\ &= 4096 - 1 \\ &= 4095 \end{aligned}$$

These correspond to the positive integers from 1 to 4095. Therefore, Option B is correct. Options A (4094), C (4096), and D (8191) are incorrect.

Final Answer: $\boxed{4095}$

Answer: (B)

Go Back to Question 10

Q11.

Solution

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

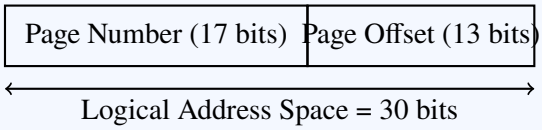
(a) **Page Number (p):** Used as an index into the page table to find the corresponding physical frame number.

(b) **Page Offset (d):** Identifies the exact byte location within the selected page.

The number of bits assigned to the page offset (d) is determined solely by the size of the page (S_{page}):

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

The remaining bits in the logical address space are used to represent the page number.



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

$$\begin{aligned} S_{\text{page}} &= 8 \text{ KB} \\ &= 8 \times 1024 \text{ Bytes} \\ &= 2^3 \times 2^{10} \text{ Bytes} = 2^{13} \text{ Bytes} \end{aligned}$$

Step 2: Determine the number of bits required to address any of the 2^{13} bytes in a page:

$$d = \log_2(2^{13}) = 13 \text{ bits}$$

Step 3: Analyze the logical address division: With a 30-bit logical address space:

- **Page Offset bits (d):** 13 bits
- **Page Number bits (p):** Logical Address Size $- d = 30 - 13 = 17$ bits

Since 13 bits are used for the page offset, Option D is correct. Options A (10), B (11), and C (12) are incorrect.

Final Answer: 13

Answer: (D)

Go Back to Question 11

Q12.

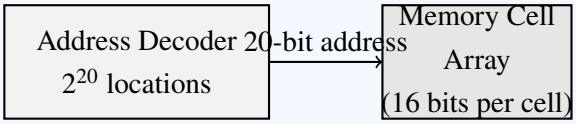
Solution

Concept: A memory module organization of the form $W \times B$ indicates:

- W: The number of addressable memory locations (words).
- B: The width of each memory location (word size) in bits.

The total storage capacity of the module in bits is $W \times B$. To convert this value into bytes, we divide the total number of bits by 8 (since 1 Byte = 8 bits):

$$\text{Capacity in Bytes} = \frac{W \times B}{8}$$



Solution: Step 1: Identify the components of the given organization $2^{20} \times 16$:

- Number of locations (W): 2^{20}
- Word size (B): 16 bits

Step 2: Calculate the total capacity in bits:

$$\text{Total Bits} = 2^{20} \times 16 \text{ bits}$$

Step 3: Convert the total bits into bytes:

$$\begin{aligned}\text{Total Bytes} &= \frac{2^{20} \times 16}{8} \text{ Bytes} \\ &= 2^{20} \times 2 \text{ Bytes}\end{aligned}$$

Step 4: Convert bytes into Megabytes (MB): Recall that 1 MB = 2^{20} Bytes (or $1,048,576$ bytes). Therefore:

$$\text{Total Bytes} = 2 \times 2^{20} \text{ Bytes} = 2 \text{ MB}$$

Thus, the storage capacity of the module is 2 MB. Options A (1 MB), C (4 MB), and D (16 MB) do not represent the correct capacity. Hence, Option B is correct.

Final Answer: 2 MB

Answer: (B)

Go Back to Question 12

Q13.

Solution

Concept: In a virtual memory system using paging, physical memory (RAM) is divided into fixed-size blocks called frames, and logical memory is divided into blocks of the same size called pages.

When a process references a logical address, the Memory Management Unit (MMU) checks the page table to translate the virtual address to a physical address. The page table entry contains a valid/invalid bit:

- **Valid:** The page is loaded in physical RAM.
- **Invalid:** The page is currently not present in physical RAM (it is stored on the secondary storage, such as a disk swap space).

If the translation process encounters an invalid bit, a hardware interrupt called a **page fault** is raised, prompting the operating system to load the page into main memory.

Solution: Step 1: Evaluate the definition of a page fault: A page fault occurs when a program attempts to access a portion of its address space that is mapped into virtual memory but is not currently loaded in the physical main memory.

Step 2: Contrast this with the options:

- **Option A (A page is modified):** Modifying a page sets the "dirty bit" in the page table, which indicates the page must be written back to storage when evicted, but this does not trigger a page fault.
- **Option B (A page is accessed for the first time):** This may cause a page fault if it is not in memory, but subsequent accesses will also cause page faults if the page gets evicted. First access alone is not the definition.
- **Option C (The required page is not present in main memory):** This is the exact condition that triggers the hardware interrupt (page fault exception).
- **Option D (Cache memory becomes full):** This leads to cache eviction algorithms, not a virtual memory page fault.

Thus, Option C is correct.

Final Answer: C

Answer: (C)

Go Back to Question 13

Q14.

Solution

Concept: Hard disk capacity is calculated using:

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

where Cylinders, Heads, Sectors per Track, and Bytes per Sector are the disk's physical storage parameters.

Solution: Step 1: Identify the given disk parameters:

- Cylinders = 4096 = 2^{12}
- Heads = 32 = 2^5
- Sectors per Track = 128 = 2^7
- Bytes per Sector = 512 = 2^9

Step 2: Multiply the values together using power-of-2 arithmetic to find the total capacity in bytes:

$$\begin{aligned}\text{Total Capacity} &= 4096 \times 32 \times 128 \times 512 \\ &= 2^{12} \times 2^5 \times 2^7 \times 2^9 \text{ Bytes} \\ &= 2^{12+5+7+9} \text{ Bytes} \\ &= 2^{33} \text{ Bytes}\end{aligned}$$

Step 3: Convert bytes into Gigabytes (GB): Recall that in binary storage measurements:

$$1 \text{ GB} = 2^{30} \text{ Bytes}$$

Using this definition, we can rewrite the capacity:

$$\begin{aligned}2^{33} \text{ Bytes} &= 2^3 \times 2^{30} \text{ Bytes} \\ &= 8 \times 1 \text{ GB} = 8 \text{ GB}\end{aligned}$$

Thus, the disk can store approximately 8 GB of data. Options A (4 GB), C (16 GB), and D (32 GB) are incorrect.

Final Answer: 8 GB

Answer: (B)

Go Back to Question 14

Q15.

Solution

Concept: Boolean expressions can be simplified using basic axioms and laws of Boolean algebra, such as:

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

	$B=0$	$B=1$
$A=0$	0	0
$A=1$	0	1

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

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

Step 2: Simplify the product of the first two terms $(A+B)(A+\overline{B})$: Using the distributive law of addition over multiplication, $(X+Y)(X+Z) = X + YZ$, we can treat A as X, B as Y, and $\overline{B}$ as Z:

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

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

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

Step 3: Substitute this simplified term back into the original expression:

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

Step 4: Distribute A over the terms inside the parentheses:

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

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

$$F = 0 + AB = AB$$

The simplified expression is AB, which matches Option A. Options B $(A+B)$, C (A), and D (B) are incorrect.

Final Answer: AB

Answer: (A)

Go Back to Question 15

Q16.

Solution

Concept: The duality principle in Boolean algebra states that any true Boolean identity remains valid if the operators and identity elements are systematically swapped according to the following rules:

- Swap the OR operator (+) with the AND operator (·).
- Swap the AND operator (·) with the OR operator (+).
- Swap the identity element 0 with the identity element 1, and vice versa.
- **Crucial Note:** The variables themselves and their complements (e.g., A or $\overline{A}$) remain completely unchanged.

Solution: Step 1: Write down the original Boolean expression:

$$E = A + BC$$

Using explicit operator notation, this expression is:

$$E = A + (B \cdot C)$$

Step 2: Apply the duality transformation rules:

- Replace the principal OR operator (+) with an AND operator (·):

$$+ \rightarrow \cdot$$

- Replace the implicit AND operator (·) between B and C with an OR operator (+):

$$\cdot \rightarrow +$$

Step 3: Re-assemble the expression maintaining the correct precedence (using parentheses):

$$\begin{aligned} E_{\text{dual}} &= A \cdot (B + C) \\ &= A(B + C) \end{aligned}$$

This matches Option A. Option B ($AB + C$), Option C ($(A + B)C$), and Option D ($AB + AC$) represent incorrect transformations.

Final Answer: A(B+C)

Answer: (A)

Go Back to Question 16

Q17.

Solution

Concept: A minterm is a product (AND) of all the variables in a Boolean function, in either direct or complemented form. For a Boolean function with n variables:

- Each variable can appear in exactly one of two states: uncomplemented (e.g., X) or complemented (e.g., $\overline{X}$).
- The total number of distinct combinations of these variables (which is equivalent to the number of rows in the truth table, or the maximum possible number of minterms) is given by:

$$\text{Total Minterms} = 2^n$$

Solution: Step 1: Identify the number of variables in the given function:

$$n = 7$$

Step 2: Calculate the maximum possible number of minterms:

$$\begin{aligned}\text{Total Minterms} &= 2^n \\ &= 2^7\end{aligned}$$

Step 3: Evaluate the exponent:

$$2^7 = 128$$

Thus, there are exactly 128 unique minterms (ranging from m_0 to m_{127}) for a 7-variable function. Options A (64), C (256), and D (512) represent functions with 6, 8, and 9 variables respectively. Thus, Option B is correct.

Final Answer: $\boxed{128}$

Answer: (B)

Go Back to Question 17

Q18.

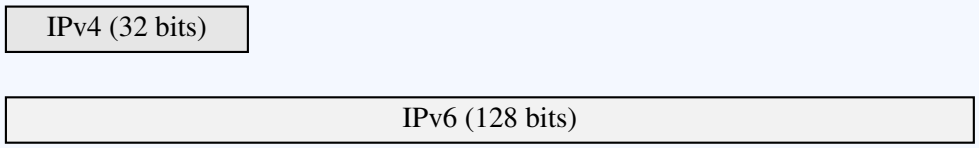
Solution

Concept: The size of an address space represents the total number of unique addresses that can be expressed with a given bit width. For an address length of b bits, the total number of unique addressable locations is:

$$\text{Address Space Size} = 2^b$$

To determine how many times larger address space Y is compared to address space X, we calculate the ratio of their capacities:

$$\text{Ratio} = \frac{\text{Capacity}_Y}{\text{Capacity}_X} = \frac{2^{b_Y}}{2^{b_X}} = 2^{b_Y - b_X}$$



Solution: Step 1: State the address sizes of both protocols:

- **IPv4 address length (b_{IPv4}):** 32 bits
- **IPv6 address length (b_{IPv6}):** 128 bits

Step 2: Express the size of each address space:

$$\text{IPv4 Address Space} = 2^{32}$$
$$\text{IPv6 Address Space} = 2^{128}$$

Step 3: Calculate the relative scale factor of IPv6 compared to IPv4:

$$\begin{aligned}\text{Ratio} &= \frac{\text{IPv6 Address Space}}{\text{IPv4 Address Space}} \\ &= \frac{2^{128}}{2^{32}} \\ &= 2^{128-32} \\ &= 2^{96}\end{aligned}$$

The IPv6 address space is 2^{96} times larger than the IPv4 address space. Options A (2^{32}), B (2^{64}), and D (2^{128}) are incorrect.

Final Answer: $\boxed{2^{96}}$

Answer: (C)

Go Back to Question 18

Q19.

Solution

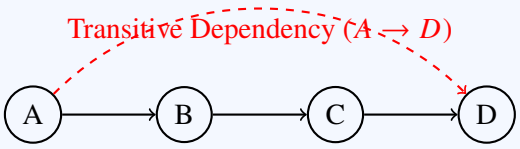
Concept: In relational database design, a functional dependency $X \rightarrow Y$ is a **transitive dependency** if it is indirect. Formally, a transitive dependency exists if there is a set of attributes Z such that:

(a) $X \rightarrow Z$ holds

(b) $Z \rightarrow Y$ holds

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

Under Armstrong's axioms, if $X \rightarrow Z$ and $Z \rightarrow Y$, then we can transitively infer $X \rightarrow Y$.



Solution: Step 1: Identify the candidate key and attribute categories:

- **Relation attributes:** $\{A, B, C, D\}$
- **Candidate Key:** A (meaning $A \rightarrow \{A, B, C, D\}$)
- **Prime Attributes:** $\{A\}$
- **Non-Prime Attributes:** $\{B, C, D\}$

Step 2: Analyze the given dependencies:

- $A \rightarrow B$ is a direct dependency from the primary key A to the non-prime attribute B.
- $B \rightarrow C$ is a direct dependency.
- $C \rightarrow D$ is a direct dependency.

Step 3: Apply the transitivity axiom: We have the chain $A \rightarrow B \rightarrow C \rightarrow D$:

- Since $A \rightarrow B$ and $B \rightarrow C$, we transitively infer $A \rightarrow C$.
- Since $A \rightarrow C$ and $C \rightarrow D$, we transitively infer $A \rightarrow D$.

Step 4: Match with the options: The dependency $A \rightarrow D$ is not in the set of base dependencies but is a derived, indirect relationship resulting from transitivity. Thus, $A \rightarrow D$ demonstrates a transitive dependency.
Option C is the correct answer.

Final Answer: $\boxed{A \rightarrow D}$

Answer: (C)

Go Back to Question 19

Q20.

Solution

Concept: Operating systems handle deadlocks using different strategies: prevention, avoidance, detection, and recovery.

Deadlock Avoidance requires the operating system to be given additional information in advance regarding which resources a process will request and use during its lifetime. Using this information, the OS can decide for each request whether the process should wait to ensure the system remains in a **safe state** (a state where there exists at least one execution sequence that allows all processes to finish without deadlock).

The classic algorithm used for deadlock avoidance in systems with multiple instances of resource types is the Banker's Algorithm, formulated by Edsger Dijkstra.

Solution: Step 1: Analyze the specific function of each algorithm listed in the options:

- **Option A (FIFO - First-In, First-Out):** Used for basic CPU scheduling or page replacement, not deadlock handling.
- **Option B (Round Robin):** A preemptive CPU scheduling algorithm designed for time-sharing systems.
- **Option C (Banker's Algorithm):** Specifically designed to evaluate resource requests in a system with N processes and M resource types to dynamically avoid deadlocks by keeping the system in a safe state.
- **Option D (LRU - Least Recently Used):** A page replacement algorithm used in virtual memory management.

Step 2: Match with the problem statement: The problem specifies a system with 5 processes and 3 resource types seeking dynamic deadlock avoidance, which is the exact use case for the Banker's Algorithm. Thus, Option C is correct.

Final Answer: Banker's Algorithm

Answer: (C)

Go Back to Question 20

Answer Key

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