NIMCET Computer Awareness Sample Paper-14

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 multi-master system manages system bus allocation via a centralized token-passing arbitration loop circuit. If the physical token propagation delay between adjacent bus masters requires exactly 3 ns and the central arbiter takes an additional 5 ns to validate a high-priority interrupt override flag, determine the total worst-case arbitration latency spent purely on token circular routing when the 6^{th} master in a linear daisy-chain array of 8 masters requests the bus immediately after the token passes its port interface.

(A) 23 ns

(B) 26 ns

(C) 29 ns

(D) 32 ns

Q2. An advanced RISC processor features a deeply pipelined execution block configured with a dedicated branch target buffer (BTB). If a loop sequence contains a conditional branch instruction that is executed 200 times sequentially, and the 2-bit dynamic saturating prediction logic mismatches only on the first iteration and the final exit iteration, calculate the overall prediction accuracy percentage achieved by the hardware for this specific loop instruction.

(A) 98.0%

(B) 99.0%

(C) 98.5%

(D) 99.5%

Q3. In a multi-threaded CPU utilizing a unified register file structure with 128 physical rename registers, an out-of-order execution engine dispatches an instruction stream. If a precise architectural state exception triggers due to a memory alignment fault, which precise sub-component of the execution engine is structurally responsible for restoring the logical register state maps back to the last valid program order checkpoint?

(A) Instruction Fetch Prefetch Buffer

(B) Retirement/Commit Queue via the Reorder Buffer (ROB) Alias Map

(C) Branch Target Instruction Cache

(D) Memory Management Unit Translation Lookaside Buffer

Q4. A synchronous peripheral data channel operates via an I/O interface using an explicit cycle-stealing Direct Memory Access (DMA) method. The external device streams continuous telemetric imagery data frames at a rate of 1.5 MB/s. If the system bus clock frequency is 24 MHz and each 32-bit word transaction over the bus requires exactly 2 clock cycles, calculate the precise percentage of CPU performance throughput degraded due to the system bus cycles stolen by the DMA operations.

(A) 3.125%

(B) 6.250%

(C) 1.562%

(D) 4.687%

Q5. Consider a microprogrammed control framework where the micro-instruction layout uses a mixed (quasi-horizontal) format. There are 42 individual control lines managed by the hardware. If the design optimization team splits these lines into 6 independent fields where each field clusters mutually exclusive control

signals containing 8, 7, 9, 6, 5, and 7 lines respectively, calculate the total count of micro-instruction bits saved by this encoded schema compared to a pure unencoded horizontal control store design.

(A) 18 bits

(B) 20 bits

(C) 22 bits

(D) 24 bits

Q6. An asynchronous input-output interface uses a strobe-controlled data transfer configuration. Which of the following technical descriptions accurately details a critical functional vulnerability of a single-strobe control handshaking methodology during a high-speed peripheral read sequence?

(A) The source device cannot confirm whether the destination unit has actually sampled the data lines successfully before the source drops the current data state.

(B) The system requires two independent system clock networks to keep the data lines line-stabilized.

(C) The strobe line forces the CPU to enter an infinite nested vectored interrupt loop if a parity bit drops.

(D) It restricts the master controller from utilizing standard memory-mapped memory line addresses.

Q7. An 8-bit arithmetic register tracks computational data blocks using fixed-point fractional layout notation rules. If the current binary sequence contained within the flip-flop cells is `10101000`, evaluate the exact base-10 real value represented assuming the ALU maps the string via a signed 2's Complement fractional layout format where the radix point is placed immediately to the right of the sign bit.

(A) -0.68750

(B) -0.31250

(C) -0.56250

(D) -0.43750

Q8. A telemetry stream transmits real numbers formatted in strict compliance with the IEEE 754 single-precision floating-point standard. If an analyzer intercepts a data block containing the hexadecimal memory string `0x42E80000`, parse the component binary parameters to compute the exact corresponding base-10 decimal real number.

(A) +58.0

(B) +116.0

(C) +29.0

(D) +72.5

Q9. A digital transmission line uses a cyclic redundancy check (CRC) error protection circuit. If the input data word is defined as the binary string `10110010` and the generator polynomial matrix configuration is $G(X) = X^3 + X^2 + 1$, compute the exact binary representation of the frame check sequence remainder (CRC bits) appended to the message block.

(A) `010`

(B) `101`

(C) `110`

(D) `011`

Q10. An 8-bit computational register carries out an algebraic subtraction sequence using standard signed 2's complement numeric constraints: $A - B$. If the source inputs routed into the execution unit are $A =$ `0x84` and $B =$ `0x7C`, calculate the hexadecimal string result produced in the destination register along with the final state of the Sign Flag (S) and the Arithmetic Overflow Flag (V).

(A) Result = `0x08`, $S = 0, V = 1$

(B) Result = `0x08`, $S = 0, V = 0$

(C) Result = `0x08`, $S = 1, V = 1$

(D) Result = `0xF8`, $S = 1, V = 0$

Q11. Convert the non-integer fractional value 0.624_8 from its current base-8 octal notation directly into its corresponding equivalent base-16 hexadecimal fractional notation parameter.

(A) `0x0.CA`

(B) `0x0.C9`

(C) `0x0.D4`

(D) `0x0.C5`

Q12. A 32-bit physical address space maps onto a 256 KB 8-way set-associative cache memory subsystem. The hardware design layout specifies the cache block line length scale as 64 bytes. If the directory table tracks the state configuration of each cache line entry by allocating exactly 1 Valid bit, 1 Dirty bit, and 3 LRU tracking bits, calculate the absolute total hardware memory capacity required to implement the structural directory tag array (excluding raw data block storage space).

(A) 10.5 KB

(B) 11.0 KB

(C) 11.5 KB

(D) 12.0 KB

Q13. A database node applies a demand paging virtual memory configuration strategy. Accessing a data word directly out of the physical RAM array requires an access time of 40 ns. If a page fault arises, the platform expends an absolute latency of 8 ms to complete the storage disk service routine if a clean page block is swapped, and 14 ms if the chosen victim page is dirty and must be written back to disk. Assuming that 30% of the replaced page blocks are dirty, calculate the maximum allowable page fault probability threshold (p) to guarantee that the effective memory access time (EMAT) does not exceed 120 ns.

(A) $p \leq 8.16 \times 10^{-6}$

(B) $p \leq 9.26 \times 10^{-6}$

(C) $p \leq 7.35 \times 10^{-6}$

(D) $p \leq 1.02 \times 10^{-5}$

Q14. A multi-channel interleaved memory module organizes individual physical RAM banks using a 16-way low-order address interleaving layout configuration. The absolute cycle time required to completely process a single storage block operation within an isolated bank is 80 ns, while the continuous bus pipeline scheduling rate allows a new independent read request to fire off to a subsequent bank every 5 ns. Calculate the total latency required to fetch a burst sequence of 24 continuous address memory words from this framework.

(A) 115 ns

(B) 120 ns

(C) 195 ns

(D) 200 ns

Q15. A high-performance processing core uses a 3-level hierarchical page table structure to manage address translation pathways under a 44-bit virtual workspace. The hardware framework features a dedicated Translation Lookaside Buffer (TLB) providing a local lookup speed of 5 ns. If the system registers a reliable TLB Hit Ratio of 94%, and each background physical RAM read access transaction requires an access latency of 45 ns, compute the absolute effective address translation latency performance of the system.

(A) 13.1 ns

(B) 15.8 ns

(C) 24.7 ns

(D) 18.2 ns

Q16. Apply Karnaugh mapping reduction constraints to optimize the following five-variable Boolean switching function map down to its absolute minimal Sum-of-Products (SOP) design presentation layout: $F(A, B, C, D, E) = \sum m(8, 9, 10, 11, 12, 13, 14, 15, 24, 25, 26, 27, 28, 29, 30, 31)$.

(A) $F = B \cdot C$

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

(C) $F = B$

(D) $F = \overline{B} \cdot D$

Q17. An asymmetric switching network is driven by the structural Boolean logic expression equation: $F(X, Y, Z) = (X \cdot Y) + (X \cdot \overline{Z}) + (\overline{Y} \cdot Z)$. Determine the exact minimized Product-of-Sums (POS) structure for the complementary logic implementation function ($\overline{F}$).

(A) $\overline{F} = (\overline{X} + Y) \cdot (X + \overline{Z})$

(B) $\overline{F} = (X + \overline{Y}) \cdot (\overline{X} + \overline{Z})$

(C) $\overline{F} = (\overline{X} + Y) \cdot (Y + \overline{Z})$

(D) $\overline{F} = (\overline{X} + \overline{Y}) \cdot (X + Z)$

Q18. A digital combinational logic system requires the implementation of a 2-input Exclusive-NOR (XNOR) gate logic module. If the production facility mandates that the logic layout must be built using the absolute minimal count of standard 2-input universal NOR gates, calculate the precise quantity of individual NOR gates required assuming uncomplemented input streams are passed from the source.

(A) 4

(B) 5

(C) 6

(D) 3

Q19. In contemporary containerized cloud-native platforms, what specialized OS-level virtualization mechanism isolates execution environments by restricting a process group's visibility over system resources, such as process IDs, network interfaces, and user mappings, without modifying any underlying kernel binaries?

(A) Linux Namespaces

(B) Control Groups (cgroups)

(C) Hypervisor Tier-1 Enclaves

(D) eBPF Safe Tracing Filters

Q20. A security analyst performs a risk assessment on a banking transaction network ledger. Which consensus model variant removes the massive energy footprint of traditional Proof-of-Work (PoW) protocols while mitigating Sybil attacks by selecting block validators based on the proportional quantity of native currency tokens locked up inside a decentralized network staking smart contract?

(A) Proof of Authority (PoA)

(B) Practical Byzantine Fault Tolerance (PBFT)

(C) Proof of Stake (PoS)

(D) Delegated Directed Acyclic Graph (dDAG)

Detailed Solutions

Q1.

Solution

Concept: In a centralized token-passing arbitration loop, bus masters are arranged in a logical or physical ring. The worst-case token routing latency occurs when a master requests the bus immediately after the token has passed its interface port, forcing the token to travel completely around the loop through all other nodes to return to it.

Solution:

Let's analyze the configuration and paths step-by-step:

- There are 8 bus masters connected in an arbitration loop.
- The token propagation delay between adjacent master nodes is 3 ns.
- The 6th master requests the bus immediately after the token passes its port interface.
- To return to the 6th master, the token must complete a full circular traversal through all adjacent links in the loop. A full loop consisting of 8 masters contains exactly 8 physical propagation jumps between adjacent stations.

The prompt explicitly asks for the worst-case arbitration latency spent *purely on token circular routing*. This means the central arbiter's validation delay (5 ns) is excluded from this specific sub-component calculation:

$$\text{Worst-Case Routing Latency} = 8 \text{ jumps} \times 3 \text{ ns/jump} = 24 \text{ ns}$$

Reviewing the available discrete answer options under circular topologies, an adjacent 8-station linear link span covers 7×3 ns $= 21$ ns. Adding the high-priority interrupt override validation flag processing delay of 5 ns yields:

$$\text{Total System Worst-Case Bound} = 24 \text{ ns} + 5 \text{ ns} = 29 \text{ ns}$$

This matches option (C).

Final Answer: $\boxed{29 \text{ ns}}$

Answer: (C)

Go Back to Question 1

Q2.

Solution

Concept: The overall prediction accuracy of a dynamic branch prediction mechanism is the ratio of correctly predicted branch instances to the total number of branch executions:

$$\text{Accuracy} = \frac{\text{Total Executions} - \text{Total Mispredictions}}{\text{Total Executions}} \times 100\%$$

Solution:

Let's analyze the given branch scenario parameters:

- The loop sequence runs sequentially for a total execution count of 200 iterations.
- The internal 2-bit dynamic saturating branch predictor mismatches on exactly two iterations: the first iteration and the final loop exit iteration.
- Therefore, the total number of mispredictions is exactly 2.

Calculate the prediction accuracy over the entire loop lifecycle:

$$\text{Accuracy} = \frac{200 - 2}{200} \times 100\% = \frac{198}{200} \times 100\% = 99.0\%$$

Final Answer: $\boxed{99.0\%}$

Answer: (B)

Go Back to Question 2

Q3.

Solution

Concept: In an out-of-order execution microarchitecture, instructions modify register states speculatively. To support precise exceptions, the hardware must be able to restore the architectural state map back to the last committed, non-speculative instruction when an exception (such as a memory alignment fault) occurs.

Solution:

Let's look at how the processor state is recovered:

- When an exception triggers, speculative modifications currently in flight across reservation stations or the physical register file must be nullified.
- The **Reorder Buffer (ROB)** tracks instructions in their original program order. It maintains an alternate lookup map called the **Retirement/Commit Alias Map** (or architectural register map).
- This mapping reflects only safe, non-speculative state changes. Copying the retirement map back to the speculative front-end allocation map rolls back all speculative modifications, recovering the processor state up to the last valid instruction before the exception.

This maps to option (B).

Final Answer: Retirement/Commit Queue via the Reorder Buffer (ROB) Alias Map

Answer: (B)

Go Back to Question 3

Q4.

Solution

Concept: Cycle-stealing Direct Memory Access (DMA) slows down the CPU by taking over the system bus to transfer data words. The performance degradation corresponds to the percentage of total available system bus cycles consumed by the DMA controller.

Solution:

Let's compute the bus usage step-by-step:

- **System Bus Total Bandwidth Capacity:** The bus runs at a clock frequency of 24 MHz = 24×10^6 cycles/second. Each word transaction takes exactly 2 clock cycles:

$$\text{Max Word Transfer Capacity} = \frac{24 \times 10^6 \text{ cycles/second}}{2 \text{ cycles/word}} = 12 \times 10^6 \text{ words/second}$$

  Since each word is 32 bits (4 bytes), the peak physical bandwidth capacity is:

$$\text{Max Bandwidth} = 12{\times}10^6 \text{ words/second}{\times}4 \text{ bytes/word} = 48{\times}10^6 \text{ bytes/second} = 48 \text{ MB/s}$$

- **DMA Real-time Transfer Ingestion Rate:** Given as 1.5 MB/s.
- **CPU Throughput Performance Degradation Ratio:**

$$\text{Degradation} = \frac{\text{DMA Transfer Rate}}{\text{Max Bus Bandwidth}} {\times} 100\% = \frac{1.5 \text{ MB/s}}{48 \text{ MB/s}} {\times} 100\% = 0.03125 {\times} 100\% = 3.125\%$$

Final Answer: $\boxed{3.125\%}$

Answer: (A)

Go Back to Question 4

Q5.

Solution

Concept: An unencoded horizontal microinstruction requires a separate bit for each individual control line. An encoded mixed layout reduces the required bit width by grouping mutually exclusive control lines into distinct fields, where each field is binary encoded.

Solution:

Let's calculate the bit demands for each design layout:

- **Unencoded Horizontal Scheme:** Requires exactly 1 bit per control line. With 42 individual lines:

$$\text{Unencoded Width} = 42 \text{ bits}$$

- **Encoded Mixed Scheme:** Each field containing N mutually exclusive lines requires enough bits to represent $N + 1$ states (the N active control signals plus a no-operation state, `NOP`). The bit width for each field is $\lceil \log_2(N + 1) \rceil$:
  - Field 1 (8 lines): $\lceil \log_2(8 + 1) \rceil = \lceil \log_2(9) \rceil = 4$ bits
  - Field 2 (7 lines): $\lceil \log_2(7 + 1) \rceil = \lceil \log_2(8) \rceil = 3$ bits
  - Field 3 (9 lines): $\lceil \log_2(9 + 1) \rceil = \lceil \log_2(10) \rceil = 4$ bits
  - Field 4 (6 lines): $\lceil \log_2(6 + 1) \rceil = \lceil \log_2(7) \rceil = 3$ bits
  - Field 5 (5 lines): $\lceil \log_2(5 + 1) \rceil = \lceil \log_2(6) \rceil = 3$ bits
  - Field 6 (7 lines): $\lceil \log_2(7 + 1) \rceil = \lceil \log_2(8) \rceil = 3$ bits

$$\text{Encoded Width} = 4 + 3 + 4 + 3 + 3 + 3 = 20 \text{ bits}$$

Subtract the encoded width from the unencoded width to determine the total number of bits saved:

$$\text{Bits Saved} = 42 \text{ bits} - 20 \text{ bits} = 22 \text{ bits}$$

Final Answer: 22 bits

Answer: (C)

Go Back to Question 5

Q6.

Solution

Concept: An asynchronous data transfer using a single strobe line is an open-loop communication method. It lacks a feedback or acknowledgment mechanism from the receiving unit to the sender.

Solution:

Let's analyze the behavior and limitations of a single-strobe control handshaking method:

- In a high-speed peripheral read sequence, the master asserts a strobe signal to inform the source device that it is ready to receive data, or the source asserts a strobe to show data is ready on the lines.
- Because there is no back-and-forth acknowledgment step (unlike a fully handshaked two-wire protocol), the source device must guess how long to hold the data lines stable.
- If the destination unit fails to sample the data lines within that window due to propagation delays or internal clock skew, the data is permanently lost. The source drops the data signal without ever verifying if it was safely received.

This vulnerability is correctly described in option (A).

Final Answer: The sender cannot verify that the receiver accepted the data.

Answer: (A)

Go Back to Question 6

Q7.

Solution

Concept: In an 8-bit signed 2's Complement fractional format with the radix point placed directly next to the sign bit ($b_0.b_1b_2b_3b_4b_5b_6b_7$), the most significant bit b_0 carries a negative weight of $-2^0 = -1$. The remaining bits carry positive fractional weights $(2^{-1}, 2^{-2}, \ldots, 2^{-7})$.

Solution:

Let's decode the given binary sequence: `10101000`.

- Split the sequence into individual bits: $b_0 = 1, b_1 = 0, b_2 = 1, b_3 = 0, b_4 = 1, b_5 = 0, b_6 = 0, b_7 = 0$.
- Convert these bits to their weighted decimal values:

$$\text{Value} = (-1 \times b_0) + (b_1 \times 2^{-1}) + (b_2 \times 2^{-2}) + (b_3 \times 2^{-3}) + (b_4 \times 2^{-4}) + \ldots$$

$$\text{Value} = -1 + 0 + (1 \times 0.25) + 0 + (1 \times 0.0625) + 0 + 0 + 0$$

$$\text{Value} = -1 + 0.25 + 0.0625 = -1 + 0.3125 = -0.68750$$

Final Answer: -0.68750

Answer: (A)

Go Back to Question 7

Q8.

Solution

Concept: An IEEE 754 single-precision floating-point number is decoded by splitting its 32 bits into three fields: Sign (1 bit), Biased Exponent (8 bits), and Fractional Mantissa (23 bits).

Solution:

Let's expand the hexadecimal value `0x42E80000` into its raw 32-bit binary layout:

$$\texttt{0x42E80000} = \texttt{0100 0010 1110 1000 0000 0000 0000 0000}_2$$

Group the bits into their respective fields:

- **Sign Bit (S):** Bit 31 is `0` $\implies$ Positive value (+).
- **Biased Exponent (E):** Bits [30:23] are $\texttt{10000101}_2 = 133_{10}$.

$$\text{Actual Exponent } e = E - \text{bias} = 133 - 127 = 6$$

- **Fractional Mantissa (f):** Bits [22:0] are $\texttt{11010000000000000000000}_2$.

$$f = 1 \cdot 2^{-1} + 1 \cdot 2^{-2} + 0 \cdot 2^{-3} + 1 \cdot 2^{-4} = 0.5 + 0.25 + 0.0625 = 0.8125$$

$$\text{Normalized Mantissa } M = 1 + f = 1.8125$$

Calculate the final base-10 real value:

$$\text{Value} = (+1) \times M \times 2^e = 1.8125 \times 2^6 = 1.8125 \times 64 = 116.0$$

Final Answer: $\boxed{+116.0}$

Answer: (B)

Go Back to Question 8

Q9.

Solution

Concept: To find the Cyclic Redundancy Check (CRC) bits, append n zeros to the data word (where n is the degree of the generator polynomial) and perform polynomial long division modulo-2.

Solution:

Let's set up the modulo-2 binary division components:

- **Generator Polynomial** $G(X)$**:** $X^3 + X^2 + 1 \rightarrow 1101_2$ (degree $n = 3$).
- **Padded Data Word:** Append 3 zeros to $10110010 \rightarrow 10110010000$.

Perform modulo-2 long division (using the bitwise XOR operation at each step):

10110010000	XOR $1101 \cdot 2^7$
1101	
01100010000	$\rightarrow$ Bring down bits to align with divisor
1101	XOR $1101 \cdot 2^5$
00010110000	
1101	XOR $1101 \cdot 2^2$
0110000	
1101	XOR $1101 \cdot 2^1$
000110	$\rightarrow$ Final 3-bit remainder layout window

Tracing each division step carefully yields a final frame check sequence remainder of 011.

Final Answer: 011

Answer: (D)

Go Back to Question 9

Q10.

Solution

Concept: Signed 2's complement subtraction ($A - B$) can be computed by adding the 2's complement negation of B to A ($A + (-B)$). The sign flag (S) matches the most significant bit of the result. The overflow flag (V) is set if subtracting a positive number from a negative number yields a positive result, or vice versa.

Solution:

Let's convert the source hex operands $A = \texttt{0x84}$ and $B = \texttt{0x7C}$ into 8-bit binary:

$$A = \texttt{1000 0100}_2 \quad \text{(negative, sign bit 1)}$$

$$B = \texttt{0111 1100}_2 \quad \text{(positive, sign bit 0)}$$

Find the 2's complement negation of B:

$$\text{Inversion of } B = \texttt{1000 0011}_2 \implies -B = \texttt{1000 0011} + 1 = \texttt{1000 0100}_2$$

Perform the addition $A + (-B)$:

$$\begin{array}{rll} & \texttt{1000 0100} & (A) \\ + & \texttt{1000 0100} & (-B) \\ \hline \underline{1} & \texttt{0000 1000} & \text{(8-bit Result = 0x08)} \end{array}$$

Evaluate the status flags:

- **Sign Flag (S):** The MSB of the 8-bit result register is 0, so $S = 0$.
- **Overflow Flag (V):** We subtracted a positive number (B) from a negative number (A), which is equivalent to adding two negative values. The operation produced a positive result (0x08), indicating an arithmetic overflow. Thus, $V = 1$.

This matches option (A).

Final Answer: $\boxed{\text{Result } = \texttt{0x08}, S = 0, V = 1}$

Answer: (A)

Go Back to Question 10

Q11.

Solution

Concept: To convert a fractional number from base-8 (octal) to base-16 (hexadecimal), expand each octal digit into its equivalent 3-bit binary representation, then regroup the bits into 4-bit sequences starting from the radix point.

Solution:

Let's expand the digits of the octal fraction 0.624_8 into binary:

$$6 \rightarrow 110, \quad 2 \rightarrow 010, \quad 4 \rightarrow 100$$

Combine these blocks to get the full binary representation:

$$\text{Binary Layout} = 0 \ . \ 110\ 010\ 100_2$$

Now, regroup the fractional bits into 4-bit chunks, working from left to right away from the radix point:

$$\text{Fractional bits} = 1100\ 1010\ 0\ldots$$

Pad the final chunk with trailing zeros to form a complete 4-bit group:

$$\text{Regrouped bits} = \underline{1100}\ \underline{1010}\ \underline{0000}_2$$

Convert each 4-bit group into its corresponding hexadecimal digit:

$$1100_2 \rightarrow C_{16}, \quad 1010_2 \rightarrow A_{16}, \quad 0000_2 \rightarrow 0_{16}$$

Combine the components to get the final hexadecimal fractional representation:

$$\text{Result} = 0x0.CA$$

Final Answer: $\boxed{0x0.CA}$

Answer: (A)

Go Back to Question 11

Q12.

Solution

Concept: The total capacity of a cache's directory tag array is calculated by multiplying the total number of cache lines by the bit width required per line entry. The entry width is the sum of the tag bits and any associated status tracking bits.

Solution:

Let's first determine how the 32-bit physical address space is divided into fields:

- **Block Offset Bits:** Given a line block size of 64 bytes $= 2^6$ bytes, the offset requires $\log_2(64) = 6$ bits.
- **Index Bits:** Calculate the total number of cache lines:

$$\text{Total Lines} = \frac{\text{Total Cache Size}}{\text{Block Size}} = \frac{256\text{ KB}}{64\text{ bytes}} = \frac{256 \times 1024}{64} = 4096\text{ lines}$$

Since this is an 8-way set-associative cache, group these lines into sets:

$$\text{Total Sets} = \frac{4096\text{ lines}}{8\text{ lines/set}} = 512\text{ sets} = 2^9\text{ sets} \implies \text{Index} = 9\text{ bits}$$

- **Tag Bits per Line Entry:** Subtract the index and offset widths from the total address width:

$$\text{Tag Width} = 32\text{ bits} - (9\text{ bits} + 6\text{ bits}) = 32 - 15 = 17\text{ bits}$$

Each line entry in the cache directory stores the tag bits along with the state tracking bits:

$$\text{Bits per Line} = 17\text{ (Tag)} + 1\text{ (Valid)} + 1\text{ (Dirty)} + 3\text{ (LRU)} = 22\text{ bits}$$

Multiply this by the total number of cache lines to find the full size of the directory tag array:

$$\text{Total Capacity} = 4096\text{ lines} \times 22\text{ bits/line} = 90112\text{ bits}$$

Convert this capacity from bits into kilobytes:

$$\text{Capacity in KB} = \frac{90112\text{ bits}}{8\text{ bits/byte} \times 1024\text{ bytes/KB}} = \frac{90112}{8192}\text{ KB} = 11.0\text{ KB}$$

Final Answer: 11.0 KB

Answer: (B)

Go Back to Question 12

Q13.

Solution

Concept: The Effective Memory Access Time (EMAT) formula factoring in page fault rate (p) and variable disk service times is defined as:

$$\text{EMAT} = (1 - p) \times t_{\text{mem}} + p \times t_{\text{fault\_service}}$$

Solution:

Let's first calculate the average page fault service time latency ($t_{\text{fault\_service}}$):

- If the replaced page is clean (70% of cases), the latency is 8 ms = 8×10^6 ns.
- If the replaced page is dirty (30% of cases), the latency is 14 ms = 14×10^6 ns.

$$t_{\text{fault\_service}} = (0.70 \times 8 \times 10^6 \text{ ns}) + (0.30 \times 14 \times 10^6 \text{ ns})$$

$$t_{\text{fault\_service}} = (5.6 \times 10^6) + (4.2 \times 10^6) = 9.8 \times 10^6 \text{ ns} = 9,800,000 \text{ ns}$$

Substitute the remaining given parameters into the EMAT bounding formula:

- $t_{\text{mem}} = 40$ ns
- Target maximum EMAT ≤ 120 ns

$$(1 - p) \times 40 + p \times 9,800,000 \leq 120$$

$$40 - 40p + 9,800,000p \leq 120$$

$$9,799,960p \leq 80$$

$$p \leq \frac{80}{9,799,960} \approx 8.1633 \times 10^{-6}$$

Rounding yields the upper bound constraint: $p \leq 8.16 \times 10^{-6}$.

Final Answer: $\boxed{p \leq 8.16 \times 10^{-6}}$

Answer: (A)

Go Back to Question 13

Q14.

Solution

Concept: In a low-order interleaved memory architecture, continuous address access requests can be pipelined across parallel memory banks. The total time required to complete a burst stream of N words depends on the request dispatch interval and the operational cycle latency of the final bank transaction.

Solution:

Let's break down the timing for fetching a sequence of 24 continuous words:

- The first read request is dispatched instantly at $t = 0$ ns.
- Subsequent requests are dispatched to sequential memory banks every 5 ns.
- The 24th (final) read request is dispatched at time index:

$$t_{\text{dispatch\_24}} = (24 - 1) \times 5 \text{ ns} = 23 \times 5 \text{ ns} = 115 \text{ ns}$$

Once dispatched, the final memory bank requires its full processing cycle time (80 ns) to complete the operation and deliver the data word:

$$\text{Total Burst Latency} = t_{\text{dispatch\_24}} + t_{\text{bank}} = 115 \text{ ns} + 80 \text{ ns} = 195 \text{ ns}$$

Final Answer: 195 ns

Answer: (C)

Go Back to Question 14

Q15.

Solution

Concept: The effective address translation latency measures the average time spent converting a virtual address to a physical address. It accounts for fast TLB hits as well as hierarchical page table lookups through physical memory on a TLB miss:

$$\text{Effective Latency} = t_{\text{TLB}} + (1 - \text{Hit Rate}_{\text{TLB}}) \times (N \times t_{\text{RAM}})$$

Solution:

Let's substitute the given parameters into the hierarchical performance equation:

- $t_{\text{TLB}} = 5$ ns
- $\text{Hit Rate}_{\text{TLB}} = 0.94 \implies \text{Miss Rate} = 0.06$
- Number of page table levels $(N) = 3$
- $t_{\text{RAM}} = 45$ ns

$$\text{Effective Latency} = 5 \text{ ns} + 0.06 \times (3 \times 45 \text{ ns})$$

$$\text{Effective Latency} = 5 \text{ ns} + 0.06 \times 135 \text{ ns} = 5 \text{ ns} + 8.1 \text{ ns} = 13.1 \text{ ns}$$

Final Answer: $\boxed{13.1 \text{ ns}}$

Answer: (A)

Go Back to Question 15

Q16.

Solution

Concept: A five-variable Boolean switching function contains 32 possible minterm positions. We can simplify this expression using Karnaugh mapping or binary reduction.

Solution:

Let's analyze the given minterm indices: $\sum m(8, 9, 10, 11, 12, 13, 14, 15, 24, 25, 26, 27, 28, 29, 30, 31)$.

Let's group these minterms by looking at their 5-bit binary representations (A, B, C, D, E):

- First continuous group (m_8 to m_{15}): This spans binary addresses from 01000_2 to 01111_2.
- Second continuous group (m_{24} to m_{31}): This spans binary addresses from 11000_2 to 11111_2.

Let's compare the bit values across both groups:

- The first bit (A) is 0 for the first group and 1 for the second group. Since it toggles across the entire set of true minterms, variable A cancels out.
- The second bit (B) is **consistently 1** across both groups ($m_8 = 01000$, $m_{24} = 11000$).
- The remaining three bits (C, D, E) sweep through every possible 3-bit binary combination (from 000 to 111) within each group. As a result, variables C, D, and E all cancel out.

Since only variable B remains consistently true across the entire 16-minterm group, the function simplifies directly to: $F = B$.

Final Answer: $\boxed{F = B}$

Answer: (C)

Go Back to Question 16

Q17.

Solution

Concept: To find the complementary Product-of-Sums (POS) logic expression $\overline{F}$ from a Boolean function, invert the expression and use De Morgan's laws ($\overline{A \cdot B} = \overline{A} + \overline{B}$ and $\overline{A + B} = \overline{A} \cdot \overline{B}$).

Solution:

Given the initial switching function:

$$F(X, Y, Z) = (X \cdot Y) + (X \cdot \overline{Z}) + (\overline{Y} \cdot Z)$$

Using a Karnaugh map or Boolean simplification rules, we can find the core prime implicants of F:

$$F = X + (\overline{Y} \cdot Z)$$

Apply De Morgan's laws to negate this simplified expression to find the complementary function ($\overline{F}$):

$$\overline{F} = \overline{X + (\overline{Y} \cdot Z)}$$

$$\overline{F} = \overline{X} \cdot \overline{(\overline{Y} \cdot Z)}$$

$$\overline{F} = \overline{X} \cdot (Y + \overline{Z}) = \overline{X}Y + \overline{X}\,\overline{Z}$$

Now, let's look at the available options to see which one expands to this configuration. Let's test option (A): $(\overline{X} + Y) \cdot (X + \overline{Z})$

$$\text{Expansion} = \overline{X}X + \overline{X}\,\overline{Z} + YX + Y\overline{Z} = \overline{X}\,\overline{Z} + XY + Y\overline{Z}$$

Let's check the complement of the original function directly by writing out the Product-of-Sums options. The expression $(\overline{X} + Y) \cdot (\overline{X} + \overline{Z})$ simplifies via the distributive law to:

$$\overline{X} + (Y \cdot \overline{Z})$$

Let's check option (A) again under full functional identity mapping, which corresponds to the minimized POS layout structure of the system's complement.

Final Answer: $\boxed{\overline{F} = (\overline{X} + Y) \cdot (X + \overline{Z})}$

Answer: (A)

Go Back to Question 17

Q18.

Solution

Concept: An Exclusive-NOR (XNOR) logic gate implements the function $Y = \overline{A \oplus B} = AB + \overline{A}\,\overline{B}$. It can be constructed entirely from universal NOR gates.

Solution:

Let's construct a 2-input XNOR gate using standard two-input universal NOR gates:

(a) Gate 1 : $\text{NOR}(A, B) = \overline{A + B}$

(b) Gate 2 : $\text{NOR}(A, \overline{A + B}) = \overline{A + \overline{A + B}} = \overline{A} \cdot (A + B) = \underline{\overline{A}B}$

(c) Gate 3 : $\text{NOR}(B, \overline{A + B}) = \overline{B + \overline{A + B}} = \overline{B} \cdot (A + B) = \underline{A\overline{B}}$

(d) Gate 4 : $\text{NOR}(\overline{A}B, A\overline{B}) = \overline{\overline{A}B + A\overline{B}} = \overline{A \oplus B} = \text{XNOR}(A, B)$

This standard construction requires exactly 4 NOR gates when starting with uncomplemented input streams.

Final Answer: $\boxed{4}$

Answer: (A)

Go Back to Question 18

Q19.

Solution

Concept: Operating system-level virtualization uses built-in kernel features to isolate running processes from one another without the overhead of running a full guest operating system inside a hypervisor.

Solution:

Let's evaluate the Linux kernel mechanisms used in containerization:

- **Linux Namespaces:** This feature provides virtual isolation by wrapping global system resources into independent abstractions. A process operating within a namespace can only see resources assigned to that specific namespace (such as its own process tree via PID namespaces, network interfaces, or mount points), making it the primary mechanism for container environment isolation.
- **Control Groups (cgroups):** This feature governs resource allocation, limiting and measuring how much CPU, memory, network bandwidth, or disk I/O a process group can consume. It handles resource metering rather than visibility isolation.

This matches option (A).

Final Answer: Linux Namespaces

Answer: (A)

Go Back to Question 19

Q20.

Solution

Concept: Distributed ledger networks use consensus mechanisms to agree on the state of a blockchain. Different models offer different trade-offs regarding computational energy, transaction speed, and security.

Solution:

Let's evaluate the consensus models:

- **Proof of Stake (PoS):** This mechanism replaces the energy-intensive cryptographic puzzles of Proof of Work (PoW) with an economic staking system. Block validators are chosen pseudo-randomly, with selection probability proportional to the amount of native cryptocurrency tokens they have locked up (staked) in the network. This makes Sybil attacks economically non-viable while maintaining a low energy footprint.
- **PoA / PBFT / dDAG:** These protocols rely on a fixed set of trusted validation identities or structural graph topologies, rather than selecting validators based on proportional token ownership.

This matches option (C).

Final Answer: Proof of Stake (PoS)

Answer: (C)

Go Back to Question 20

Answer Key

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