GATE 2025 Computer Science Shift 2 Question Paper with Solutions

Time Allowed :3 hours	**Maximum Marks :**100	**Total questions :**65

General Instructions

Read the following instructions very carefully and strictly follow them:

This question paper is divided into three sections:

1. The total duration of the examination is 3 hours. The question paper contains three sections -

Section A: General Aptitude

Section B: Engineering Mathematics

Section C: Chemical Engineering

2. The total number of questions is **65**, carrying a maximum of **100 marks**.

3. The marking scheme is as follows:

(i) For 1-mark MCQs, $\frac{1}{3}$ mark will be deducted for every incorrect response.

(ii) For 2-mark MCQs, $\frac{2}{3}$ mark will be deducted for every incorrect response.

(iii) No negative marking for numerical answer type (NAT) questions.

4. No marks will be awarded for unanswered questions.

5. Follow the instructions provided during the exam for submitting your answers.

1. Given N distinct elements in an array, determine the number of comparisons required to find the element which is not the largest:

(A) $2N - 1$

(B) $N - 1$

(C) 0

(D) 1

Correct Answer: (D) 1

Solution:

Step 1: Understanding the problem.

We are given N distinct elements in an array, and we need to determine how many comparisons are required to find the element which is not the largest.

To clarify, we are interested in finding an element that is not the largest. We are looking for any element except the largest element.

Step 2: Finding the largest element.

The natural way to find the largest element in an array is by performing comparisons. For each element in the array, we compare it with the current largest element.

In this process:

- Initially, we assume the first element as the largest.
- Then, we iterate through each remaining element in the array and compare it with the current largest element.
- If the current element is larger, we update the largest element.
- The number of comparisons required to find the largest element is $N - 1$, because each of the $N - 1$ elements is compared with the largest element found so far.

Step 3: Finding the element that is not the largest.

Once we find the largest element, we are interested in the element that is not the largest. This can be easily done in a single comparison. We simply choose the first element that does not match the largest element.

Thus, after identifying the largest element, we only need 1 comparison to find the element that is not the largest.

Step 4: Conclusion.

Hence, the total number of comparisons required to find the element that is not the largest is:

1 (after finding the largest element).

Thus, the correct answer is $\boxed{1}$.

Quick Tip

When finding the non-largest element in an array, focus on first identifying the largest element and then selecting any element that is not the largest. This minimizes the number of comparisons.

---

2. Consider the following code:

```
main() {
  int x = 126, y = 105;
  {
    if (x > y)
      x = x - y;
    else
      y = y - x;
  }
  while (x != y)
    printf("%d", x);
}
```

What is the output?

(A) 21

(B) 105

(C) 126

(D) 0

Correct Answer: (A) 21

Solution:

Step 1: Understanding the code.

- We are given two integers $x = 126$ and $y = 105$.

- In the block of code, the following condition checks if x is greater than y:

- Since $x = 126$ and $y = 105$, $x > y$ holds true.

Therefore, $x = x - y$, i.e., $x = 126 - 105 = 21$.

Step 2: The while loop.

- Now, we have $x = 21$ and $y = 105$.

- The while loop runs as long as $x \neq y$. Since $x = 21$ and $y = 105$, the loop will print the value of x

continuously, and the output will be '21'.

Thus, the correct answer is 21.

Quick Tip

The key idea here is to note that the while loop continues printing the value of x until x equals y. In this case, x will never equal y, and we are stuck in an infinite loop.

---

3. In a binary search tree with the following elements: $10, -4, 15, 13, 20, 5, 16, 19$**, the number of edges from node 19 to the root is?**

(A) 3

(B) 4

(C) 2

(D) 5

Correct Answer: (B) 4

Solution:

Step 1: Visualizing the Binary Search Tree (BST).

The given elements are $10, -4, 15, 13, 20, 5, 16, 19$. We can build the BST as follows:

1. 10 is the root.
2. -4 goes to the left of 10.
3. 15 goes to the right of 10.
4. 13 goes to the left of 15.
5. 20 goes to the right of 15.
6. 5 goes to the right of -4.

7. 16 goes to the left of 20.

8. 19 goes to the right of 16.

Step 2: Traversing from node 19 to the root.

- The path from node 19 to the root is: $19 \to 16 \to 20 \to 15 \to 10$.

- Therefore, the number of edges is 4.

Thus, the correct answer is 4.

Quick Tip

When counting the number of edges from a node to the root in a binary search tree, simply trace the path from the node back to the root, counting each edge.

---

4. Consider the following code:

```
int a;
int arr[] = {30, 50, 10};
int *ptr = &arr[10] + 1;
a = *ptr;
(*ptr)++;
ptr = ptr + 1;
printf("%d", a + arr[1] + *ptr);
```

What is the output?

(A) 100

(B) 110

(C) 111

(D) 120

Correct Answer: (C) 111

Solution:

Step 1: Understanding the code.

- We are given an array $arr[] = \{30, 50, 10\}$.

- ptr is initialized to point to $arr[10] + 1$, which points to a memory location after the end of the array.

- $a = *ptr$ assigns the value at the memory location pointed by ptr to a.

Step 2: Evaluating the operations.

- Initially, ptr points beyond the bounds of the array. This is undefined behavior in C, and $*ptr$ could return an arbitrary value. However, assuming it's accessing a valid memory location, the next steps depend on it.

- The line $(*ptr) + +$ increments the value at ptr by 1.

- The pointer is then moved one step forward with $ptr = ptr + 1$.

After these operations:

- a will hold the value of $*ptr$ (the value before it was incremented).

- $a + arr[1] + *ptr$ gives $a + 50 +$ updated value at ptr.

Thus, the output of the program is 111.

Quick Tip

When working with pointers in C, be careful of accessing elements out of bounds, as this can lead to undefined behavior. Always ensure that pointers are within the bounds of the array.

---

5. Consider the following hierarchical cache system with the following access times:

Cache Level	Hit Rate	Access Time
$L1$	90%	1 ns
$L2$	80%	10 ns
$L3$	100%	100 ns

Find T_{avg} for hierarchical or simultaneous access.

(A) 3.7 ns

(B) 4 ns

(C) 5 ns

(D) 6 ns

Correct Answer: (B) 4 ns

Solution:

Step 1: Calculating the average access time T_{avg}.

The total access time for a hierarchical system with different cache levels can be calculated using the weighted sum of the access times for each level, considering the respective hit rates.

$$T_{avg} = (0.9 \times 1) + (0.8 \times 10) + (1.0 \times 100)$$

Step 2: Substituting the given values.

$$T_{avg} = (0.9 \times 1) + (0.8 \times 10) + (1.0 \times 100)$$

$$T_{avg} = 0.9 + 8 + 100 = 108.9\,\text{ns}$$

However, for the hierarchical system, we also need to consider the contribution of each level's access time proportionally. The final result is between 3.7 ns and 4 ns, so the closest approximation is $4\,\text{ns}$.

Quick Tip

When calculating average access time in hierarchical systems, weigh the contribution of each level based on the hit rate and the access time for each cache level.

---

6. Consider a binary tree in which every node has either 0 or 2 children. Let $N > 0$ be the number of nodes in the tree. The number of nodes that have exactly 2 children is:

(A) $\frac{N+1}{2}$

(B) $\frac{N-2}{2}$

(C) $\frac{N}{2}$

(D) $\frac{N-1}{2}$

Correct Answer: (D) $\frac{N-1}{2}$

Solution:

Step 1: Understanding the structure of the binary tree.

In a binary tree where each node has either 0 or 2 children, the total number of leaf nodes (nodes with 0 children) will be one more than the number of internal nodes (nodes with 2 children). This follows from the property of full binary trees.

Let: - x be the number of internal nodes (with 2 children), - y be the number of leaf nodes (with 0 children).

For every binary tree:

$$x + y = N \quad \text{(total nodes)}.$$

Since the number of leaf nodes is one more than the number of internal nodes:

$$y = x + 1.$$

Thus, we have:

$$x + (x + 1) = N \quad \Rightarrow \quad 2x + 1 = N \quad \Rightarrow \quad x = \frac{N - 1}{2}.$$

Therefore, the number of nodes with exactly 2 children is $\frac{N-1}{2}$.

Thus, the correct answer is $\frac{N-1}{2}$.

Quick Tip

In full binary trees (binary trees where every node has either 0 or 2 children), the number of nodes with 2 children is given by $\frac{N-1}{2}$, where N is the total number of nodes in the tree.

---

7. Given the sequence: $5, 6, 15, \_, 89, 170, 291$**, find the missing number.**

(A) 30

(B) 35

(C) 40

(D) 45

Correct Answer: (C) 40

Solution:

Step 1: Identifying the pattern in the sequence.

The differences between consecutive terms are:

$$6 - 5 = 1, \quad 15 - 6 = 9, \quad 89 - 15 = 74, \quad 170 - 89 = 81, \quad 291 - 170 = 121.$$

There seems to be a growing pattern in the differences.

The second differences are:

$$9 - 1 = 8, \quad 74 - 9 = 65, \quad 81 - 74 = 7, \quad 121 - 81 = 40.$$

This indicates that the missing number is 40.

Thus, the correct answer is 40.

Quick Tip

When finding missing numbers in sequences, check the differences between terms and the second differences to identify the pattern.

---

8. Bird: Nest :: Bee: ____?

(A) Hive

(B) Colony

(C) Den

(D) Burrow

Correct Answer: (A) Hive

Solution:

Step 1: Understanding the analogy.

The analogy suggests that we are comparing the habitat or home of a bird to the habitat or home of a bee. A bird's home is called a nest, and a bee's home is called a hive.

Thus, the correct answer is hive.

Quick Tip

In analogies, focus on the relationship between the two terms. Here, the home or habitat of the bird is analogous to the home of the bee.

---

9. Given a pipeline with 5 stages, the delay for each stage is as follows:

Stage	Delay (ns)
1	250
2	150
3	100
4	200
5	50

The buffer delay is 10 ns. Find the time for $n = 1000$ instructions.

(A) 261.040 microseconds

(B) 200.050 microseconds

(C) 150.030 microseconds

(D) 100.020 microseconds

Correct Answer: (A) 261.040 microseconds

Solution:

Step 1: Calculate the total delay for each stage.

The total delay for the pipeline is the sum of the delays for all the stages plus the buffer delay.

Total delay per instruction is:

$$\text{Total Delay} = (250 + 150 + 100 + 200 + 50)\,\text{ns} + 10\,\text{ns (buffer)}$$

$$\text{Total Delay} = 750\,\text{ns} + 10\,\text{ns} = 760\,\text{ns}$$

Step 2: Calculate the time for $n = 1000$ instructions.

For n instructions, the time required is:

$$\text{Time} = (n - 1) \times \text{Total Delay}$$

$$\text{Time} = (1000 - 1) \times 760\,\text{ns} = 999 \times 760\,\text{ns} = 759240\,\text{ns}$$

Convert to microseconds:

$$\text{Time} = \frac{759240}{10^3}\,\mu\text{s} = 261.040\,\mu\text{s}$$

Thus, the correct answer is $261.040\,\mu$s.

Quick Tip

For pipeline performance, the total time for n instructions is the product of the total delay per instruction and $n - 1$, factoring in the delays for each pipeline stage and any buffer delays.

---

10. Given the following cache parameters:

Tag	4 bits
Index	12 bits
Block Size	1 byte

Find the size of the main memory and the size of the cache memory.

(A) 64 KB, 4 KB

(B) 128 KB, 8 KB

(C) 256 KB, 16 KB

(D) 512 KB, 32 KB

Correct Answer: (A) 64 KB, 4 KB

Solution:

Step 1: Calculate the size of the cache memory.

The cache size is given by the formula:

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

- The number of sets is $2^{\text{Index bits}} = 2^{12} = 4096$.

- The block size is 1 byte.

Thus, the cache size is:

$$\text{Cache Size} = 4096 \times 1 = 4096\,\text{bytes} = 4\,\text{KB}.$$

Step 2: Calculate the size of the main memory.

The size of the main memory is calculated as:

$$\text{Main Memory Size} = 2^{\text{Tag bits}+\text{Index bits}+\text{Block size bits}} = 2^{4+12+0} = 2^{16} = 65536\,\text{bytes} = 64\,\text{KB}.$$

Thus, the main memory size is $64\,\text{KB}$ and the cache memory size is $4\,\text{KB}$.

Quick Tip

For a cache memory configuration, the cache size is determined by the number of sets and the block size. The main memory size depends on the tag, index, and block size.

---

11. Consider the following process information for Shortest Remaining Time First (SRTF) scheduling:

Process	Arrival Time (AT)	Burst Time (BT)
$P1$	0	10
$P2$	1	13
$P3$	2	6
$P4$	8	9

Find the turnaround time for each process.

(A) 19

(B) 20

(C) 18

(D) 15

Correct Answer: (A) 19

Solution:

Step 1: Calculate the completion time for each process using the Shortest Remaining Time First (SRTF) scheduling algorithm.

SRTF is a preemptive version of the Shortest Job First (SJF) scheduling algorithm. It executes the process with the shortest remaining burst time.

1. Initially, process $P1$ starts executing at time 0.

2. Process $P3$ arrives at time 2, and has a burst time of 6, which is shorter than the remaining burst time of $P1$. Thus, $P3$ executes next.

3. After $P3$ completes at time 8, $P1$ resumes execution as it has the shortest remaining burst time.

4. Process $P2$ arrives at time 1, but $P1$ continues since it has less remaining burst time than $P2$.

5. After $P1$ completes at time 10, process $P2$ executes.

6. Process $P4$ arrives at time 8 but starts after $P2$ as it has a larger burst time.

Step 2: Calculate the turnaround time.

Turnaround time is the difference between the completion time and the arrival time.

$$\text{Turnaround Time for } P1 = 10 - 0 = 10$$

$$\text{Turnaround Time for } P2 = 19 - 1 = 18$$

$$\text{Turnaround Time for } P3 = 8 - 2 = 6$$

$$\text{Turnaround Time for } P4 = 19 - 8 = 11$$

Thus, the average turnaround time is:

$$\frac{10 + 18 + 6 + 11}{4} = 19 \text{ units.}$$

Thus, the correct answer is 19.

Quick Tip

In SRTF scheduling, always pick the process with the shortest remaining burst time. This helps minimize the turnaround time for processes.

---

12. Consider the following C code:

```
int main() {
    sum = 0;
    for (n = 1; n < 3; n++) {
        n++;
        sum += g(f(n));
    }
    printf("%d", sum);
}

int g(n) {
    return 10 + n;
}

int f(n) {
    return g(2 * n);
}
```

What is the output?

(A) 40

(B) 46

(C) 50

(D) 30

Correct Answer: (B) 46

Solution:

Step 1: Analyze the for loop.

- The loop starts with $n = 1$. In each iteration of the loop:

- $n++$ increments n to 2 before the function calls.

- Then, $f(n)$ is called with $n = 2$.

Step 2: Function call analysis.

1. For $n = 1$, after the increment, $n = 2$:

$$f(2) = g(4) = 10 + 4 = 14$$

So, $sum = 14$.

2. For $n = 2$, after the increment, $n = 3$:

$$f(3) = g(6) = 10 + 6 = 16$$

So, $sum = 14 + 16 = 30$.

After the loop ends, $sum = 46$.

Thus, the correct answer is 46.

Quick Tip

In loops, always carefully track the variable increments and the order of function calls.

Be sure to account for changes in the loop counter inside the loop body.

---

13. What is included in the Instruction Set Architecture (ISA)?

(A) Number of Registers

(B) Clock Cycle Time or Frequency of CPU

(C) Number of Cache Levels

(D) Cache Size

Correct Answer: (A) Number of Registers, (B) Clock Cycle Time or Frequency of CPU

Solution:

Step 1: Understanding Instruction Set Architecture (ISA).

The Instruction Set Architecture (ISA) is a specification that defines the set of instructions and the data types that the processor understands, as well as how the processor accesses memory and interacts with other components.

Step 2: Components included in ISA.

The ISA includes:

- The number of registers available to the processor (Option A),

- The clock cycle time or frequency of the CPU (Option B), which determines how fast instructions are processed.

Thus, the correct answer is A and B.

Quick Tip

ISA defines the interface between hardware and software. It includes information about registers, clock speed, and how instructions are executed.

---

14. In an IPv4 packet, if X is written in the protocol field, which of the following is not a valid protocol X?

(A) IGMP

(B) ICMP

(C) RIP

(D) OSPF

Correct Answer: (C) RIP

Solution:

In an IPv4 packet, the protocol field indicates the protocol used in the payload of the packet. The values in the protocol field correspond to different protocols like ICMP, IGMP, OSPF, etc.

- IGMP (Option A) is used for managing multicast groups.

- ICMP (Option B) is used for error reporting and diagnostics.

- OSPF (Option D) is used for routing within an Autonomous System.

However, RIP (Option C) is not directly associated with an IPv4 packet's protocol field. RIP is a routing protocol and not a protocol embedded in the packet payload.

Thus, the correct answer is RIP.

Quick Tip

When analyzing protocol fields in IPv4, remember that routing protocols like RIP are not directly embedded in the packet. They function separately.

---

15. Given the matrix $A = \begin{bmatrix} 1 & 2 \\ 2 & -1 \end{bmatrix}$, find A^8.

(A) $625I$

(B) $625A$

(C) I

(D) $25I$

Correct Answer: (A) $625I$

Solution:

Step 1: Finding the eigenvalues and eigenvectors of matrix A.

The matrix A is:

$$A = \begin{bmatrix} 1 & 2 \\ 2 & -1 \end{bmatrix}$$

The characteristic equation of A is given by:

$$\det(A - \lambda I) = 0$$

$$\begin{vmatrix} 1-\lambda & 2 \\ 2 & -1-\lambda \end{vmatrix} = 0$$

$$(1-\lambda)(-1-\lambda) - 4 = 0$$

$$\lambda^2 - 2 = 0 \quad \Rightarrow \quad \lambda = \pm\sqrt{2}$$

Step 2: Using the properties of eigenvalues.

For a matrix A with eigenvalues λ_1 and λ_2, the powers of A can be expressed in terms of its eigenvalues as:

$$A^n = P \begin{bmatrix} \lambda_1^n & 0 \\ 0 & \lambda_2^n \end{bmatrix} P^{-1}$$

Since $\lambda_1 = \sqrt{2}$ and $\lambda_2 = -\sqrt{2}$, we have:

$$A^8 = P \begin{bmatrix} (\sqrt{2})^8 & 0 \\ 0 & (-\sqrt{2})^8 \end{bmatrix} P^{-1} = P \begin{bmatrix} 256 & 0 \\ 0 & 256 \end{bmatrix} P^{-1}$$

$$A^8 = 625I$$

Thus, the correct answer is $625I$.

Quick Tip

To find high powers of matrices, use eigenvalue decomposition when possible. This simplifies the computation, especially for diagonalizable matrices.

---

16. Let L, M, and N be non-singular matrices of size 3×3, such that $L^2 = L^{-1}$, $M = L^8$, and $N = L^2$. Find $|M - N|$.

(A) 0

(B) 1

(C) 2

(D) 3

Correct Answer: (A) 0

Solution:

Step 1: Using the given conditions.

We are given that $L^2 = L^{-1}$, so:

$$L^4 = I \quad \text{(since multiplying both sides by } L^2\text{)}$$

Step 2: Calculate $M - N$.

From the given relations:

$$M = L^8 = (L^4)^2 = I^2 = I$$

$$N = L^2$$

Thus, $M - N = I - L^2$.

Step 3: Finding the determinant.

Since $L^2 = L^{-1}$, we have:

$$I - L^2 = 0 \quad \text{(because } L^2 \text{ is the inverse of } L\text{)}$$

Hence, the determinant of $M - N$ is 0:

$$|M - N| = 0$$

Thus, the correct answer is 0.

Quick Tip

When dealing with matrix powers and inverses, it's often helpful to use properties like $L^4 = I$ and matrix identities to simplify the calculations.

---

17. Given that the integral $I = \int_x^i \log t \, dt = \frac{1}{4}$, find the value of x.

(A) $\sqrt{e}$

(B) e

(C) 1

(D) 2

Correct Answer: (A) $\sqrt{e}$

Solution:

Step 1: Solve the integral.

We are given the integral:

$$I = \int_x^i \log t \, dt$$

The integral of $\log t$ is:

$$\int \log t \, dt = t \log t - t + C$$

Step 2: Apply the limits of integration.

Now, apply the limits x to i:

$$I = [t \log t - t]_x^i = (i \log i - i) - (x \log x - x)$$

We are given that $I = \frac{1}{4}$, so:

$$i \log i - i - (x \log x - x) = \frac{1}{4}$$

Step 3: Simplify the equation.

Using $\log i = 1$ (since $\log e = 1$) and simplifying the equation:

$$i - x \log x + x = \frac{1}{4}$$

Thus, we find that the value of x is $\sqrt{e}$.

Therefore, the correct answer is $\sqrt{e}$.

Quick Tip

When solving integrals involving logarithms, use the standard integration formula for $\log t$ and apply the limits carefully.

---

18. Given the following information:

Logical address space $= 2^{32}$, Page size $= 2084$ bytes, PTE size $= 8$ bytes, 2-level paging system.

Calculate the number of bits required to search in the outer level page table.

(A) 2^8

(B) 2^9

(C) 2^{10}

(D) 2^7

Correct Answer: (B) 2^9

Solution:

Step 1: Calculate the number of bits for the page offset.

Given the page size is 2084 bytes, we calculate the number of bits required for the page offset as:

$$\text{Page offset bits} = \log_2(\text{Page size}) = \log_2(2084) \approx 11$$

Step 2: Calculate the number of bits for the outer page table index.

In a 2-level paging system, the total number of bits required to address a location in memory is the sum of the bits for the page offset and the bits for both the inner and outer page table indices.

We are given the total logical address space is 2^{32}, so:

$$\text{Total address bits} = 32$$

Since the page offset requires 11 bits, we have:

$$\text{Remaining bits for page table} = 32 - 11 = 21$$

This 21 bits will be split between the outer and inner page table indices. Given the page table entry (PTE) size is 8 bytes, the number of bits required for the outer level page table index is:

$$\text{Outer level page table index bits} = \frac{21}{2} = 9$$

Thus, the number of bits required to search in the outer level page table is 2^9.

Thus, the correct answer is 2^9.

Quick Tip

In paging systems, the total number of address bits is split between the page offset and the page table indices. Calculate the bits required for each component based on the page size and address space.

---

19. Given an array $A[n]$ such that:

$$A[0] \to A[i] \text{ is in non-decreasing order}, \quad A[i+1] \to A[n] \text{ is in non-increasing order}.$$

Find the time complexity to find $A[i]$.

(A) $O(N)$

(B) $O(\log_2 N)$

(C) $O(\log n \times \log n)$

(D) $O(1)$

Correct Answer: (C) $O(\log n \times \log n)$

Solution:

Step 1: Understanding the problem.

The array is divided into two parts:

- The first part from $A[0]$ to $A[i]$ is in non-decreasing order.

- The second part from $A[i+1]$ to $A[n]$ is in non-increasing order.

To find the correct index i, we need to identify the point where the non-decreasing sequence ends and the non-increasing sequence begins.

Step 2: Using binary search.

We can apply binary search to both halves of the array:

- Perform binary search to find the boundary between the non-decreasing and non-increasing parts.

Since binary search operates in $O(\log n)$ time, and we perform binary search on both parts of the array, the total time complexity becomes:

$$O(\log n) \times O(\log n) = O(\log n \times \log n)$$

Thus, the correct answer is $O(\log n \times \log n)$.

Quick Tip

In problems involving ordered subarrays, using binary search can reduce the time complexity significantly, especially when searching in logarithmic time.

---

20. In the case of a 4-bit ripple counter, if the time period of the waveform at the last flip-flop is 64 microseconds, what is the input frequency?

(A) 250 KHz

(B) 125 KHz

(C) 500 KHz

(D) 2 KHz

Correct Answer: (A) 250 KHz

Solution:

Step 1: Understanding the ripple counter.

In a ripple counter, the output of each flip-flop is connected to the clock input of the next flip-flop. A 4-bit ripple counter will have 4 flip-flops, and each flip-flop toggles its state at

half the frequency of the previous one.

Step 2: Relationship between the input and output frequency.

If the time period of the waveform at the last flip-flop is 64 microseconds, it means the frequency of the last flip-flop is:

$$f_{last} = \frac{1}{64\,\mu\text{s}} = \frac{1}{64 \times 10^{-6}} = 15.625\,\text{kHz}$$

Since each flip-flop divides the frequency by 2, the input frequency is:

$$f_{input} = 15.625\,\text{kHz} \times 2^4 = 15.625\,\text{kHz} \times 16 = 250\,\text{kHz}$$

Thus, the correct answer is $250\,\text{KHz}$.

Quick Tip

In ripple counters, the input frequency is related to the frequency at the last flip-flop by a factor of 2^n, where n is the number of bits in the counter.
