ASSEMBLY LANGUAGE

Section A

1. What is assembly language?
2. Define opcode and operand in machine code.
3. What is the function of an assembler?
4. List one advantage of using assembly language over machine code.
5. What does LDD do in assembly language?
6. Explain immediate addressing with an example.
7. What is the accumulator (ACC) used for in assembly language?
8. Define direct addressing.
9. What is the purpose of indexed addressing?
10. Write an assembly instruction to load the number 50 into the ACC.
11. What is the difference between absolute addressing and symbolic addressing?
12. Define indirect addressing with an example.
13. What is a trace table in program debugging?
14. Describe the use of ADD in assembly language.
15. What does the instruction JMP do?
16. Define relative addressing.
17. What does the CMP instruction do in assembly language?
18. What does the instruction INC do to a register?
19. How does a two-pass assembler differ from a single-pass assembler?
20. Write an instruction to subtract 5 from the accumulator.
21. Explain what happens during the fetch-decode-execute cycle.
22. What does the instruction OUT do?
23. Define symbolic labels in assembly language.
24. Explain how the END instruction is used.
25. What does the STO instruction do?
26. Describe a cyclic shift in binary operations.
27. What is the result of a logical shift left by two positions for 1010?

28. Write an assembly program to load two numbers, add them, and store the result.
29. What is the purpose of bit masking?
30. Define the data movement instructions.
31. What does the OR instruction achieve in bit manipulation?
32. What is the purpose of the JPE instruction?
33. Write an example program to compare ACC with 10 and jump if equal.
34. What are the types of shifts used in assembly language?
35. What does LSL perform in assembly instructions?
36. State one use of logical AND in bit manipulation.
37. How does XOR help in clearing a bit?
38. What does the relative addressing mode allow a program to do?
39. What is the role of the Index Register (IX)?
40. Why do we use labels in assembly language?
41. Define mask in bit manipulation.
42. Explain arithmetic shift right with an example.
43. What does binary addition represent in assembly instructions?
44. Write an instruction to store the ACC value at memory address 200.
45. Explain the significance of the instruction set for a CPU.
46. What does the instruction MOV achieve in assembly language?
47. Write the instruction to jump to address 100 if ACC > 0.
48. What does binary subtraction represent in assembly?
49. Write an assembly code snippet to output the value in ACC.
50. How does the assembler convert mnemonics into machine code?

Section B

1. Write a program to add three numbers and store the result.
2. Explain the steps in the two-pass assembly process.
3. How does indexed addressing simplify accessing array elements?
4. Explain the role of symbolic addressing in program readability.
5. Write an assembly program to load a number, increment it, and output the result.
6. What is the difference between immediate and indirect addressing?
7. Write an instruction to compare ACC with 15 and jump if greater.
8. Describe how trace tables are used to debug programs.
9. Write a program to compare two numbers and store the larger one.
10. What happens if a label is used but not defined in an assembly program?
11. Write a code snippet using LSL and LSR to perform multiplication and division by powers of two.
12. Describe the fetch-execute cycle for an assembly instruction.
13. Write a program to check if a bit is set using bit masking.
14. How does conditional branching help in program flow?
15. Write an instruction to jump to the next label if ACC is not equal to 0.
16. Write a simple assembly code to output all elements of an array.
17. Compare the use of ADD and SUB in program logic.
18. How does XOR clear specific bits in a register?
19. Write a code snippet to increment ACC until it reaches 10.
20. Write a program that uses CMP to check if two values are equal.
21. Explain how assembler directives aid in program execution.
22. Write a program to subtract two numbers and output the result.
23. How does indexed addressing improve looping in assembly language?
24. Write a code snippet that uses labels to manage loops.
25. What are the key differences between logical and arithmetic shifts?
26. Write a program to check if a number is even or odd using AND.
27. Explain the use of unconditional jumps in program control.
28. Write a program to multiply two numbers by repeated addition.
29. How is the symbol table used in a two-pass assembler?

30. Explain how binary shifts are useful in mathematical operations.
31. Write a program to add an array of numbers using indexed addressing.
32. How does assembly language differ from high-level languages?
33. Write a program to clear specific bits in ACC using XOR.
34. Explain the purpose of relative addressing in assembly programming.
35. Write a program to count the number of 1s in a binary value.
36. What does JPN achieve in program logic?
37. Write a program to set a specific bit in ACC.
38. Explain the use of the index register in accessing arrays.
39. Write a program to calculate the factorial of a number.
40. What does comparison and branching achieve in programming logic?
41. Explain the difference between bit masking with AND and OR.
42. Write a program to swap two numbers in memory.
43. Write a program to test if a number is greater than 100.
44. How does assembly language manage memory efficiently?
45. Write a program to toggle all bits of a binary value.
46. Explain the process of translating mnemonics into opcodes.
47. Write a program to find the maximum value in an array.
48. Write a program to reverse the bits in a binary value.
49. Explain why comments are important in assembly programs.
50. Write a code snippet to perform division using subtraction.?

Section C

1. Write a program to sort an array in ascending order using assembly.
2. Explain the role of the symbol table in resolving forward references.
3. Write a program to count the number of set bits in a byte.
4. Explain how relative addressing helps in implementing loops.
5. Write a program to multiply two numbers using shifts and adds.
6. Describe the difference between logical and arithmetic operations in bit manipulation.
7. Write a program to check if a number is prime using assembly instructions.
8. Write a program to left-align the bits in a register.
9. Explain how indexed addressing improves performance in array processing.
10. Write a program to simulate a simple calculator for addition and subtraction.
11. How can trace tables be used to validate logical instructions?
12. Write a program to convert a hexadecimal value into binary.
13. Write a program to simulate a counter that counts from 0 to 255.
14. How does the assembler resolve forward references in a two-pass assembly?
15. Write a program to check if a string is a palindrome using indexed addressing.
16. Write a program to rotate the bits in a register cyclically.
17. Explain the role of masking in assembly programming.
18. Write a program to clear all bits except the leftmost set bit.
19. How does JPE differ from JPN in conditional branching?
20. Write a program to count down from 10 to 0.
21. Write a program to reverse a string stored in memory.
22. Explain how logical shifts can be used for bit manipulation.
23. Write a program to compare two arrays for equality.
24. Describe the steps involved in debugging assembly programs.
25. Write a program to calculate the sum of even numbers in an array.
26. How does conditional branching improve program control?
27. Write a program to perform bitwise AND between two values.
28. Write a program to perform matrix addition in assembly language.

29. How does relative addressing simplify the implementation of loops?
30. Write a program to toggle every alternate bit of a byte.
31. Write a program to calculate the sum of numbers divisible by 3 in an array.
32. Explain how addressing modes influence program design.
33. Write a program to perform subtraction without using the SUB instruction.
34. Write a program to simulate a digital clock using loops and counters.
35. How does XOR clearing ensure bit accuracy in bitwise operations?
36. Write a program to implement a stack using indexed addressing.
37. Write a program to convert ASCII characters to their binary representation.
38. Explain the purpose of arithmetic shifts in signed binary numbers.
39. Write a program to find the largest power of 2 less than a given number.
40. Write a program to simulate a ring counter using cyclic shifts.
41. How does the instruction set of a CPU affect assembly programming?
42. Write a program to generate Fibonacci numbers using assembly.
43. Write a program to detect overflow in addition and subtraction.
44. Explain the significance of labels and loops in program flow.
45. Write a program to replace all negative numbers in an array with 0.
46. Write a program to compare two numbers and output the smaller one.
47. Write a program to divide two numbers without using the division instruction.
48. Explain how binary shifts are used for division by powers of two.
49. Write a program to find the most significant bit (MSB) of a number.
50. How does the fetch-decode-execute cycle handle branching instructions?