**PART 2:**

1. /\* Program that finds the largest number in a list of integers   \*/
3. .text                  // executable code follows
4. .global  \_start
5. \_start:
6. MOV      R4, #RESULT   // R4 points to result location
7. LDR      R2, [R4, #4]  // R2 holds number of elements in the list
8. MOV      R3, #NUMBERS  // R3 points to the list of integers
9. LDR      R0, [R3]      // R0 holds the largest number so far
11. LOOP:    SUBS     R2, #1        // decrement the loop counter
12. BEQ      DONE          // if result is equal to 0, branch
13. ADD      R3, #4
14. LDR      R1, [R3]      // get the next number
15. CMP      R0, R1        // check if larger number found
16. BGE      LOOP
17. MOV      R0, R1        // update the largest number
18. B        LOOP
19. DONE:    STR      R0, [R4]      // store largest number into result location
21. END:     B        END
23. RESULT:  .word    0
24. N:       .word    7             // number of entries in the list
25. NUMBERS: .word    4, 5, 3, 6    // the data
26. .word    1, 8, 2
28. .end

**Part 3:**

1. /\* Program that finds the largest number in a list of integers \*/
2. .text // executable code follows
3. .global \_start
5. \_start:
6. MOV R4, #RESULT  // R4 points to result location
7. LDR R2, [R4, #4] // R0 holds the number of elements in the list
8. MOV R1, #NUMBERS // R1 points to the start of the list
9. MOV R0, #0       //The largest number is now zero
10. BL LARGE         // Start the subroutine
11. STR R0, [R4] // R0 holds the subroutine return value
13. END: B END
15. // Subroutine for finding the largest number from a list of integers
16. LARGE:
17. SUBS   R2, #1   //Subtract 1 from the counter
18. MOVEQ PC, LR    //Return back to main is the subtraction returns zero
19. LDR   R3, [R1]  //Load the integer from the numbers list
20. ADD   R1, #4    //Add 4 to the pointer to the numbers list to get next value
21. CMP   R0, R3    //Compare the current largest value and a number from the list
22. BGE   LARGE     //If the current largest value is greater restart the loop
23. MOV   R0, R3    //Otherwise make the current largest value the list value
24. B     LARGE     //Restart the subroutine

27. RESULT:     .word 0
28. N:          .word 7 // number of entries in the list
29. NUMBERS:    .word 4, 5, 3, 6 // the data
30. .word 1, 8, 2
31. .end

**Part 4:**

1. /\* Program that converts a binary number to decimal \*/
2. .text               // executable code follows
3. .global \_start
4. \_start:
5. MOV    R4, #N
6. MOV    R5, #Digits  // R5 points to the decimal digits storage location
7. LDR    R4, [R4]     // R4 holds N
8. MOV    R0, R4       // parameter for DIVIDE goes in R0
9. BL     DIVIDE
10. STRB   R6, [R5, #3]
11. STRB   R3, [R5, #2]
12. STRB   R1, [R5, #1] // Tens digit is now in R1
13. STRB   R0, [R5]     // Ones digit is in R0
14. END:        B      END
16. /\* Subroutine to perform the integer division R0 / 10.
17. \* Returns: quotient in R1, and remainder in R0 \*/
18. DIVIDE:     MOV    R2, #0 //R2 is the tens bit
19. MOV           R3, #0 //R3 is the hundreds bit
20. MOV           R6, #0 //R6 is the thousands bit

23. THOUSANDS:  CMP    R0, #1000
24. BLT    HUNDREDS
25. SUB    R0, #1000
26. ADD    R6, #1
27. B      THOUSANDS
29. HUNDREDS:        CMP           R0, #100
30. BLT           TENS
31. SUB           R0, #100
32. ADD    R3, #1
33. B           HUNDREDS
35. TENS:                CMP           R0, #10
36. BLT           DIV\_END
37. SUB           R0, #10
38. ADD    R2, #1
39. B           TENS
41. DIV\_END:    MOV    R1, R2     // quotient in R1 (remainder in R0)
42. MOV    PC, LR

45. N:          .word  9178         // the decimal number to be converted
46. Digits:     .space 4          // storage space for the decimal digits
48. .end