```
/* Program that finds the largest number in a list of integers  */

            .text                   // executable code follows
            .global _start
_start:
            MOV     R4, #RESULT     // R4 points to result location
            LDR     R0, [R4, #4]    // R0 holds the number of elements in the list
            ADD     R1, R4, #8      // R1 points to the start of the list
            BL      LARGE
            STR     R0, [R4]        // R0 holds the subroutine return value

END:        B       END

/* Subroutine to find the largest integer in a list
 * Parameters: R0 has the number of elements in the lisst
 *             R1 has the address of the start of the list
 * Returns: R0 returns the largest item in the list
 */
LARGE:      MOV     R2, R0          // R2 now has number of elements in the list
            LDR     R0, [R1]        // R0 holds the largest number so far

LOOP:       SUBS    R2, R2, #1      // Decrement loop counter
            BEQ     DONE            // Loop ends when R2 reaches 0
            ADD     R1, #4          // Go to the next number's address
            LDR     R3, [R1]        // Get the next number
            CMP     R0, R3          // Check if larger number found
            BGE     LOOP            // If not found do not update and go to next
            iteration
            MOV     R0, R3          // Update largest number
            B       LOOP            // Go to next iteration

DONE:       MOV     pc, lr          // Return to main
// End of LARGE subroutine

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

            .end
```

```
/* Program that converts a binary number to decimal */
            .text                   // executable code follows
            .global _start
_start:
            MOV     R4, #N
            ADD     R5, R4, #4      // R5 points to the decimal digits storage location
            LDR     R4, [R4]        // R4 holds N
            MOV     R0, R4          // dividend for DIVIDE goes in R0
            MOV     R1, #1000       // Get the thousands digit
            BL      DIVIDE
            STRB    R1, [R5, #3]    // Store the thousands digit
            MOV     R1, #100        // Get the hundreds digit
            BL      DIVIDE
            STRB    R1, [R5, #2]    // Store the hundreds digit
            MOV     R1, #10         // Get the tens digit
            BL      DIVIDE
            STRB    R1, [R5, #1]    // Store the tens digit
            STRB    R0, [R5]        // Ones digit is in R0 (remainder)
END:        B       END

/* Subroutine to perform the integer division R0 / R1.
 * Parameters: dividend in R0, divisor in R1
 * Returns: quotient in R1, and remainder in R0
*/
DIVIDE:     MOV     R2, #0
CONT:       CMP     R0, R1
            BLT     DIV_END
            SUB     R0, R1
            ADD     R2, #1
            B       CONT
DIV_END:    MOV     R1, R2          // quotient in R1 (remainder in R0)
            MOV     PC, LR

N:          .word   9876              // the decimal number to be converted
Digits:     .space  4               // storage space for the decimal digits

            .end

```