```
1               .text                   // executable code follows
2               .global _start
3
4   _start:     MOV R5, #0
5               MOV R6, #0
6               MOV R7, #0
7               MOV R4, #TEST_NUM       // Load data
8   M_LOOP:     LDR R1, [R4]            // Load word into R1
9               CMP R1, #0              // Check if 0
10              BEQ DISPLAY             // End if 0
11              BL ONES                 // Else use ones subroutine
12              CMP R0, R5              // Check if new val is larger
13              MOVGT R5, R0            // If it is store it in r5
14              LDR R1, [R4]            // Load same val as before
15              BL ZEROS                // Use zeros subroutine
16              CMP R0, R6              // Check if new val is larger
17              MOVGT R6, R0            // If it is store it in r6
18              LDR R1, [R4]            // Load same val as before
19              BL ALTERNATE            // Use alternate subroutine
20              CMP R0, R7              // Check if new val is larger
21              MOVGT R7, R0            // If it is store it in r7
22              ADD R4, #4              // Move to next word
23              B M_LOOP
24
25  ONES:       MOV R0, #0              // R0 will hold the result
26  O_LOOP:     CMP R1, #0              // Loop until the data contains no more 1's
27              BEQ O_END
28              LSR R2, R1, #1          // Perform SHIFT, followed by AND
29              AND R1, R1, R2
30              ADD R0, #1              // Count the string length so far
31              B O_LOOP
32  O_END:      MOV PC, LR
33
34  ZEROS:      MOV R0, #0              // R0 will hold the result
35              MVN R1, R1              // Invert r1 to find longest string of 0's
36  Z_LOOP:     CMP R1, #0              // Loop until the data contains no more 1's
37              BEQ Z_END
38              LSR R2, R1, #1          // Perform SHIFT, followed by AND
39              AND R1, R1, R2
40              ADD R0, #1              // Count the string length so far
41              B Z_LOOP
42  Z_END:      MOV PC, LR
43
44
45  ALTERNATE:  MOV R0, #0              // Store result in r0
46              MOV R3, #ALT_NUM        // r3 = 01010101....
47              LDR R3, [R3]
48              MOV R2, R1              // Copy r1 into r2
49              EOR R1, R2, R3
50              PUSH {R2, LR}           // Store r2 and our link to main
51              BL ONES
52              POP {R2}
53              LSL R3, #1              // r3 = 10101010.....
54              EOR R1, R2, R3
55              PUSH {R0}               // Push our current r0
56              BL ONES
57              MOV R1, R0              // Move new r0 into r1
58              POP {R0}
59              CMP R1, R0              // Compare new r0 (r1) with old r0 (r0)
60              MOVGT R0, R1            // If larger then store new value
61              POP {PC}                // Return to main
62
63  DISPLAY:    LDR R8, =0xFF200020 // base address of HEX3-HEX0
64              MOV R0, R5              // display R5 on HEX1-0
65              BL DIVIDE               // ones digit will be in R0; tens digit in R1
66              MOV R9, R1              // save the tens digit
67              BL SEG7_CODE
68              MOV R4, R0              // save bit code
69              MOV R0, R9              // retrieve the tens digit, get bit code
```

```
70              BL SEG7_CODE
71              LSL R0, #8
72              ORR R4, R0
73              MOV R0, R6          // display R6 on HEX3-2
74              BL DIVIDE           // ones digit will be in R0; tens digit in R1
75              MOV R9, R1          // save the tens digit
76              BL SEG7_CODE
77              LSL r0, #16
78              ORR R4, R0          // save bit code
79              MOV R0, R9          // retrieve the tens digit, get bit code
80              BL SEG7_CODE
81              LSL R0, #24
82              ORR R4, R0
83              STR R4, [R8]        // display the numbers from R6 and R5
84              LDR R8, =0xFF200030 // base address of HEX5-HEX4
85              MOV R0, R7          // display R7 on HEX5-4
86              BL DIVIDE           // ones digit will be in R0; tens digit in R1
87              MOV R9, R1          // save the tens digit
88              BL SEG7_CODE
89              MOV R4, R0          // save bit code
90              MOV R0, R9          // retrieve the tens digit, get bit code
91              BL SEG7_CODE
92              LSL R0, #8
93              ORR R4, R0
94              STR R4, [R8]        // display the number from R7
95              B END
96
97  /* Subroutine to perform the integer division R0 / 10.
98  * Returns: quotient in R1, and remainder in R0
99  */
100 DIVIDE:     MOV R2, #0
101 CONT:       CMP R0, #10
102             BLT DIV_END
103             SUB R0, #10
104             ADD R2, #1
105             B CONT
106 DIV_END:    MOV R1, R2              // quotient in R1 (remainder in R0)
107             MOV PC, LR
108
109 /* Subroutine to convert the digits from 0 to 9 to be shown on a HEX display.
110  *    Parameters: R0 = the decimal value of the digit to be displayed
111  *    Returns: R0 = bit patterm to be written to the HEX display
112  */
113 SEG7_CODE:  MOV R1, #BIT_CODES
114             ADD R1, R0              // index into the BIT_CODES "array"
115             LDRB R0, [R1]           // load the bit pattern (to be returned)
116             MOV PC, LR
117
118 END:        B END
119
120 BIT_CODES:  .byte 0b00111111, 0b00000110, 0b01011011, 0b01001111, 0b01100110
121             .byte 0b01101101, 0b01111101, 0b00000111, 0b01111111, 0b01100111
122             .skip 2      // pad with 2 bytes to maintain word alignment
123
124 ALT_NUM:    .word 0x55555555    // 010101010101... in binary
125
126 TEST_NUM:   .word 0x103fe00f
127             .word 0x420b1a23
128             .word 0x11111111
129             .word 0x00000003
130             .word 0x00000001
131             .word 0xffffffff
132             .word 0x12345678
133             .word 0x9abcdef0
134             .word 0x42042069
135             .word 0xfedcba98
136             .word 0x00000000
137             .end
138
```