```
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 END              // 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  END:        B END
64
65  ALT_NUM:    .word 0x55555555     // 010101010101... in binary
66
67  TEST_NUM:   .word 0x103fe00f
68              .word 0x420b1a23
69              .word 0x11111111
```

```
70              .word 0x00000003
71              .word 0x00000001
72              .word 0xffffffff
73              .word 0x12345678
74              .word 0x9abcdef0
75              .word 0x42042069
76              .word 0xfedcba98
77              .word 0x00000000
78              .end
79
```