Part 2:

```
.global _start

_start: MOV R0, #0
                MOV R2, #TEST_NUM
                LDR R1, [R2]
                MOV R3, #3
                MOV R5, #0

LOOP:           SUB R3, #1
                CMP R3, #0
                BEQ END
                MOV R0, #0

                BL ONES
                LDR R1, [R2, #4]

                CMP R0, R5
                BLE LOOP
                MOV R5, R0
                B LOOP

ONES: CMP    R1, #0       // loop until the data contains no more 1's
                BEQ    ONES_OVER
                LSR    R4, R1, #1     // perform SHIFT, followed by AND
                AND    R1, R1, R4
                ADD    R0, #1        // count the string length so far
                BGE    ONES

ONES_OVER: BX LR

END:    B     END

RESULT: .word 0

TEST_NUM: .word  0x7ff, 0x1f

       .end
```

Part 3:

```
.global _start

_start: MOV R0, #0
                MOV R2, #TEST_NUM
                LDR R1, [R2]
                MOV R3, #3
                MOV R4, #0;
                MOV R5, #0
                MOV R6, #0
                MOV R7, #0

LOOP:           SUB R3, #1
                CMP R3, #0
                BEQ END
                MOV R0, #0

                //Store largest string of ones in R0
                BL ONES

                //Load R5 with the largest string of 1s
                CMP     R5, R0
                MOVLT   R5, R0

                // Store largest string of 0's in R0
                BL      ZEROES

                //Store largest # of 0's in R6
                CMP     R6, R0
                MOVLT   R6, R0

                //Store result of alternating digits in R0
                BL      ALTERNATE

                //Store result of ALTERNATE in R7
                CMP     R7, R0
                MOVLT   R7, R0

                LDR R1, [R2, #4]
                B LOOP

ONES: CMP     R1, #0          // loop until the data contains no more 1's
```

```
                BEQ    ONES_OVER
                LSR    R4, R1, #1      // perform SHIFT, followed by AND
                AND    R1, R1, R4
                ADD    R0, #1          // count the string length so far
                BGE    ONES

ONES_OVER:  MOV R4, #0;
                        BX LR

ZEROES:   PUSH    {R1, LR}  //Retain LR value because we run a subroutine in a subroutine
       MVN    R1, R1    //Invert Data
       BL     ONES      //Run ones on the inverted Data
       POP    {R1, LR}  //restore the top of the stack into a register
       MOV    PC, LR    //Return LR

END:     B      END

ALTERNATE:  PUSH    {R1, R2, R3, LR}

        MOV    R3, #ALT_DATA //Load data template from memory
        LDR    R3, [R3]
        EOR    R1, R3  //Exclusive OR R1 and R3
        BL     ONES
        MOV    R2, R0 //Store Value of Ones in R2
        BL     ZEROES
        //R0 -> Zeros
          //R2 -> Ones
        CMP    R0, R2
        BGT    ALT_END
        MOV    R0, R2

ALT_END:   POP    {R1, R2, R3, LR}
        MOV    PC, LR

RESULT: .word 0

TEST_NUM: .word  0x7ff, 0x1f

ALT_DATA: .word   0xAAAAAAAA //101010... etc -> used in alternating

       .end
```