Study Questions (Chapter 03 – Part E)

1. Question 3.15 on page 224: Write an ARM assembly language routine to count the number of 1s in a 32-bit word in r0 and return the result in r1.
2. Question 3.16 on page 224: A word consists of the bytes b4, b3, b2, b1. Write an ARM assembly language function to re-order (transpose) these bytes in the order b1, b3, b2, b4.
3. Question 3.22 on page 224: Write ARM code to implement the following C operation.

```
int s = 0;
for ( i = 0; i < 10; i++) {
s = s + i*i;)
```

4. Question 3.39 on page 225: Write an ARM assembly language program that scans a string terminated by the null byte 0x00 and copies the string from a source location pointed at by r0 to a destination pointed at by r1.

5. Write an ARM assembly language ***program*** to count the number of characters in a *null*-terminated string (`STRING1`) and store the result in memory at the location identified by label "length".
   You may want to define the data of the program as follow:

```
STRING1 DCB "This is a test string1"   ;String1
EoS1    DCB 0x00                       ;end of string1
length  DCD 0x00                       ;to store the calculated string length
```

6. Write an ARM assembly language ***program*** to concatenate two *null*-terminated strings (`STRING1` and `STRING2`) and store the result in another *null*-terminated `STRING3`. Assume that the length of `STRING1` + the length of `STRING2` ≤ 255.
   You may want to define the strings as follow:

```
STRING1 DCB "This is a test string1"   ;String1
EoS1    DCB 0x00                       ;end of string1
STRING2 DCB "This is a test string2"   ;String
EoS2    DCB 0x00                       ;end of string2
STRING3 space 0xFF
```

Exercise 1:

Write an ARM assembly language routine to count the number of 1s in a 32-bit word in r0 and return the result in r1.

code:

```
                LDR   r0, = 0x11AB003F  ;dummy value for r1 (11 ones)
                MOV   r1,#0x0           ;clear ones counter
                MOV   r2,#32            ;use r2 as the loop counter
OnesCount       MOVS  r0,r0, ROR #1     ;Repeat: rotate r0 right set flags
                ADDCS r1,r1,#1          ;if carry set increment 1s counter
                SUBS  r2,r2,#1          ;decrement loop counter
                BNE   OnesCount         ;until all bits tested
```

If this was a subroutine Count, the code might be:

```
                area test, CODE, readwrite
                ADR   sp,Stack1
                LDR   r2,=0xFFFFFFFF        ;set up dummy r2
                STR   r2,[sp]
                LDR   r0, = 0xFFAB123A      ;dummy data
                BL    Count                 ;call routine
                MOV   r3,r1                 ;read result
                NOP
                NOP

Count           STMFD sp!,{r2,lr}           ;save r2 and return on the stack
                MOV   r1,#0x0
                MOV   r2,#4
```

Let the bytes b4, b3, b2, b1 in register r1. Transposed bytes with the below program is in register r0.

eor r3,r1,r1, ror #16

bic r3,r3,#0x00FF0000

mov r0,r1,ror #8

eor r0,r0,r3,lsr #8

Instruction 1 does Ex-OR on the operands r1 and the value of right rotated by 16 bits on register r1, I.e., (swapped r1 16 bits top and 16 bits bottom halves) and stores result in r3.

Instruction 2 does bit clear, nothing but AND operation on register r3 with immediate value 0x00FF0000.

Instruction 3 does right rotate by 8 bits on register r1 and move the result to register r0.

Instruction 4 does left rotate by 8 bits on register r3 and perform Ex-OR with the left rotated value and register r0. The result is stored in r0. Thus the four bytes are re-ordered.

ARM code for the given C code :

```
func1:
        push    {r7}
        sub     sp, sp, #12
        add     r7, sp, #0
        movs    r3, #0
        str     r3, [r7, #4]
        movs    r3, #0
        str     r3, [r7]
        b       .L2
.L3:
        ldr     r3, [r7]
        mul     r3, r3, r3
        ldr     r2, [r7, #4]
        add     r3, r3, r2
        str     r3, [r7, #4]
        ldr     r3, [r7]
        adds    r3, r3, #1
        str     r3, [r7]
.L2:
        ldr     r3, [r7]
        cmp     r3, #9
        ble     .L3
        nop
        nop
        adds    r7, r7, #12
        mov     sp, r7
        ldr     r7, [sp], #4
        bx      lr
```

Program:

```
AREA scan, CODE, READWRITE
Entry
ADR r0,Str1
ADR r1,Str2
Copy LDRB r2,[r0],#1
STRB r2,[r1],#1
CMP r2,#0x00
BNE Copy
SVC #0x123456
Str1 DCB "a sample string", 0x00
Str2 SPACE 20
END
```