Study Questions (Part 5)
Sunday March 31, 2019

Covering:

- Assembly programs and functions/subroutines

As a good start, you need to answer the question at the end of Chapter 3 of the textbook (pages 224-227).

1. Question 3.15 at 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 at 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 3.21 at page 224: What are the advantages and disadvantages of the use of the ARM's BL (branch and load) subroutine call mechanism in comparison with the conventional CISC BSR (branch to subroutine) mechanism?
4. Question 3.22 at page 224: Write ARM code to implement the following C operation.

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

5. Question 3.38 at page 225: Write suitable ARM code to implement
   if x = y call PQR else call ZXY
6. Question 3.39 at 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.
7. Question 3.40 at page 224: Write a program to copy text from the location pointed at by r0 to the location pointed at by r1. The copied version must be in reverse order. Assume the string you are copying is non-null (it contains at least one character) and that it is terminated.
8. Question 3.41 at page 224: Write a program to reverse a character string with an odd number of characters pointed at by r0. Assume the string you are copying is non-null (it contains at least one character) and that it is terminated by 0x0D (the number of characters included the terminator). You must not use an additional memory buffer (i.e., you can use only registers and the string storage itself).
9. Question 3.42 at page 224: Repeat the problem in question 3.39, but when you transfer the string, remove any occurrences of the word "the"; for example, "and the man said" would become "and man said".
10. Question 3.43 at page 225: Repeat the problem in question 3.39, but when you transfer the string, reverse the sequence of any words beginning with "t"; for example, "and the man said" would become "and eht man said".
11. Question 3.54 at page 226: Explain what this fragment of code does instruction by instruction and what purpose it achieves (assuming that register r0 is the register of interest). Note that the data in r0 must not be 0 on entry.

```
     MOV r1,#0
loop MOVS r0,r0,LSL #1
     ADDCC r1,r1,#1
     BCC loop
```

12. In the following code, write an ARM assembly instruction (at location `return`) to return from the function `fun1` to the `main` program.
What are the values of r0 and r1 at the end of the program?

```
        AREA FunctionCall, CODE, READONLY
        ENTRY
main    ADR   r1,  input
        LDR   r0,[r1],#4
call    BL    fun1
        STR   r0, [r1]
LOOP    B     LOOP

fun1    ADD r0,r0,r0
return  .........

        AREA FunctionCall, DATA, READWRITE
input   DCD 0x11
output  DCD 0x00
        END
```

13. In the following code, write an ARM assembly instruction (at location `call`) to call the function `fun1`.
What are the values of r0 and r1 at the end of the program?

```
        AREA FunctionCall, CODE, READONLY
        ENTRY
main    ADR   r1,  input
        LDR   r0,[r1],#4
call    .........
        STR   r0, [r1]
LOOP    B     LOOP

fun1    ADD r0,r0,r0
return  MOV PC,LR

        AREA FunctionCall, DATA, READWRITE
input   DCD 0x11
output  DCD 0x00
        END
```

14. Write a suitable ARM code to implement the following code, where AAA, BBB, and CCC are subroutines. Note that the AAA, BBB, and CCC subroutines will not change the values of r0 and r1. However, they will change the value of the status flags.

```
if r0 == r1 call AAA else call BBB
call CCC
```

15. Write an ARM assembly language *program* counts the number of characters in a *null* terminated sting (`STRING1`) and store the result in memory at 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  DCB 0x00                       ;to store the calculated string length
```

16. Write an ARM assembly language ***program*** to concatenate two *null* terminated stings (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
```