Study Questions (Chapter 03 – Part 8)

1. Question 3.23 on page 225: What is the effect of the following addressing mode?
   STR **r0**,[r2,r3,ROR #3]!
2. Question 3.26 on page 225: What is the effect of LDR **r0**,[r5,r6, LSL #2]?
3. Question 3.28 on page 225: What effective address is generated by LDR **r0**,[r2,-r3, LSL #1]?
4. Question 3.32 on page 225: Assume that r2 contains the initial value 00001000_{16}. Explain the effect of each of the following six instructions, and give the value in r2 after each instruction executes
   a. STR r1,**[r2]**
   b. STR r1,**[r2, #8]**
   c. STR r1,**[r2, #8]!**
   d. STR r1,**[r2], #8**
   e. STR r1,**[r2, r0, LSL #8]**
5. Question 3.60 on page 227: A computer has three eight-element vectors in memory, Va, Vb, and Vc. Each element of a vector is a 32-bit word. Write the code to calculate all elements of Vc if the i^{th} element is given by
   $Vc_i = ½ (Va_i + Vb_i)$
6. Assume that r1 is a pointer to a character array, where each array element is 8-bits. This array starts from r1[0]. Write one ARM instruction that loads the value of the first element of this array to r0, if and only if the zero flag is 1.
7. Explain the following ARM assembly instruction.
   ```
   LDRBEQ r0,[r1]
   ```
8. Assume that r1 is a pointer to a character array, where each array element is 8-bits. This array starts from r1[0]. Write one ARM instruction that stores the least significant byte in r0 into the third element of this array, if and only if the zero flag is 0.
9. Explain the following ARM assembly instruction.
   ```
   STRBNE r0,[r1,#2]
   ```
10. Assume that r1 is a pointer to an integer array, where each array element is 32-bits. This array starts from r1[0]. Write one ARM instruction that loads the value of the third element of this array to r0 and to update the value of r1 to make it a point to the value that is just loaded, if and only if the carry flag is 0.
11. Explain the following ARM assembly instruction.
    ```
    LDRCC r0,[r1,#8]!
    ```
12. Assume that r1 is a pointer to an integer array, where each array element is 32-bits. This array starts from r1[0]. Write one ARM instruction that loads the value of the first element of this array to r0, and to update the value of r1 to make it a point to the third value in the array, if and only if the overflow flag is 0.

13. Explain the following ARM assembly instruction.

```
LDRVS r0,[r1],#8
```

14. Assume that r1 is a pointer to an integer array, where each array element is 32-bits. This array starts from r1[0]. Also, assume that register r2 is used as an index for this array. Write one ARM instruction that stores the value in r0 into the element number r2 of this array, and to update the value of r1 to make it a point to the value that is just stored, if and only if the negative flag is 1.

15. Explain the following ARM assembly instruction.

```
STRMI r0,[r1,r2, LSL#2]!
```

16. ARM supports many addressing modes, including immediate addressing, register-to-register, address register indirect, address register indirect with offset, address register indirect with index, auto-indexing pre-indexed, auto-indexing post-indexed, and program counter relative.
Give an ARM instruction example for each addressing mode.

17. Write a suitable ARM assembly segment of code to implement the following code, where Z is an integer array (4 bytes per element) which is located at address 0x120.

```
       for(r0 = 0; r0 <= 20; r0++)
         Z[r0] = r0;
Exit: r1 = r0;
```

18. Explain what this fragment of code does.

```
      MOV R1,#0x120
      MOV R0,#0
Loop  CMP R0,#20
      BGT Exit
      STR R0,[R1],#4
      ADD R0,R0,#1
      B   Loop
Exit  MOV R1,R0
```

19. Write a suitable ARM assembly segment of code to implement the following code, where Z is an integer array (4 bytes per element) which is located at address 0x120.

```
       for(r0 = 0; r0 <= 20; r0++)
         Z[r0] += 0x10;
Exit: r1 = r0;
```

20. Explain what this fragment of code does.

```
      MOV R1,#0x120
      MOV R0,#0
      MOV R2,#10
Loop  CMP R0,#20
      BGT Exit
      STR R2,[R1],#4
      ADD R0,R0,#1
      B   Loop
Exit  MOV R1,R0
```