Name: _____ Student ID: _____ Date: _____
Name: _____ Student ID: _____

Worksheet #7
Decision Making and Iteration

Objectives

- To understand the decision making and iteration process in ARM processor
- To create decision making instructions from branch instructions in ARM
- To create iteration instructions from branch instructions in ARM

1. Create a new project, then type and add the following code to the project.

```
        AREA  PROG7_1,  CODE,  READONLY
        ENTRY
start
        MOV   R0,  #X
        CMP   R0,  #5
        BEQ   endif
        LDR   R1,  =0x11111111
endif   LDR   R2,  =0xFFFFFFFF
loop    B       loop
        END
```

Program 7.1

2. Replace X in the program above with the value shown in Table 7.1.
3. Build and Run the program. Then observe and record the results in Table 7.1.

The values of	The results from BEQ instruction	
X	R1	R2
3		
5		
7		

Table 7.1

Suppose that we set R0 to a value that is greater than 7 or less than 3. What values will be stored in R1 and R2?

_____
_____

Why was the value of R1 not changed in case that X is set to 5?

_____
_____

Why was the value of R2 not changed in any values of X?

_____
_____

4. From Program 7.1, replace the BEQ instruction with BNE instruction as follows:

BEQ endif ➔ **BNE endif**

5. Repeat the step 2 and 3 again. Then observe and record the results in Table 7.2.

The values of	The results from BNE instruction	
X	R1	R2
3		
5		
7		

Table 7.2

Suppose that we set R0 to a value that is greater than 7 or less than 3. What values will be stored in R1 and R2?

_____

_____

Why was the value of R1 not changed in case that X is set to 5?

_____

_____

6. Create a new project, then type and add the following code to the project.

```
        AREA  PROG7_2,  CODE,  READONLY
        ENTRY
start
        MOV   R0,  #X
        CMP    R0,  #5
        BGT    endif
        LDR   R1,  =0x11111111
endif   LDR   R2,  =0xFFFFFFFF
loop    B       loop
        END
```

Program 7.2

7. Replace X in the program above with the value shown in Table 7.3.
8. Build and Run the program. Then observe and record the results in Table 7.3.

The values of	The results from BGT instruction	
X	R1	R2
3		
4		
5		
6		
7		

Table 7.3

Suppose that we set R0 to a value that is greater than 7 or less than 3. What values will be stored in R1 and R2?

(R0 > 7) R1 = _____

(R0 < 3) R1 = _____

9. From Program 7.2, replace the BGT instruction with BGE instruction as follows:

BGT endif ➔ **BGE endif**

10. Repeat the step 7 and 8 again. Then observe and record the results in Table 7.4.

The values of	The results from BGE instruction	
X	R1	R2
3		
4		
5		
6		
7		

Table 7.4

Compare the results between Table 7.3 and 7.4. Are they different? Why?

_____

_____

11. From Program 7.2, replace the BGT instruction with BLT instruction as follows:

BGT endif ➔ **BLT endif**

12. Repeat the step 7 and 8 again. Then observe and record the results in Table 7.5.

The values of	The results from BLT instruction	
X	R1	R2
3		
4		
5		
6		
7		

Table 7.5

13. From Program 7.2, replace the BGT instruction with BGE instruction as follows:

BGT endif ➔ **BLE endif**

14. Repeat the step 7 and 8 again. Then observe and record the results in Table 7.6.

The values of	The results from BLE instruction	
X	R1	R2
3		
4		
5		
6		
7		

Table 7.6

Compare the results between Table 7.5 and 7.6. Are they different? Why?

_____

_____

Can we compare two negative values in ARM? If we replace the first two instructions in program 7.2 with **MOV R0, # -10** and **CMP R0, #-5**. What do the results store in R1,R2?

_____

15. Create a new project, then type and add the following code to the project.

```
                AREA  PROG7_3,  CODE,  READONLY
                ENTRY
start
                LDR  R1,  =X
                LDR  R2,  =Y
                MOV R0,  #0
while           CMP  R1,  R2
                BGT    endwhile
                ADD  R0,  R0,  R1
                ADD  R1,  R1,  #1
                B        while
endwhile
stop            B        stop
                END
```

Program 7.3

16. Replace X and Y in the program above with the values as shown in Table 7.7.

17. Build and Run the program. Then observe and record the results in Table 7.7.

The values of		**The results from Program 7.3**
X	**Y**	**R0**
1	5	
2	7	
3	9	
1	10	

Table 7.7

Program 7.3 shows a way doing to add values from X to Y. Modify the program to compute ***n***!, where ***n*** is a non-negative integer stored in register R2. Then store the result in R0.

Hint 5! = 5 * 4 * 3 * 2 * 1 (use **MUL** instruction)

```
        AREA  Factorial,  CODE,  READONLY
        ENTRY
start
loop    B        loop
        END
```

18. Create a new project, then type and add the following code to the project.

```
        AREA  PROG7_4,  CODE,  READONLY
        ENTRY
start
        MOV   R0,  #X
        CMP    R0,  #5
        BHI     endif
        LDR   R1,  =0x11111111
endif   LDR   R2,  =0xFFFFFFFF
loop    B       loop
        END
```

Program 7.4

19. Replace X in the program above with the value shown in Table 7.8.
20. Build and Run the program. Then observe and record the results in Table 7.8.

The values of	**The results from BHI instruction**	
X	**R1**	**R2**
3		
4		
5		
6		
7		

Table 7.8

From Program 7.4, if we replace the first two instructions in the program with **MOV R0, # -10** and **CMP R0, #-5**. What do the results store in R1,R2? Are the flow of program correct? Why?

_____
_____
_____

21. From Program 7.4, replace the BHI instruction with BHS instruction as follows:

BHI endif ➔ **BHS endif**

22. Repeat the step 19 and 20 again. Then observe and record the results in Table 7.9.

The values of	**The results from BHS instruction**	
X	**R1**	**R2**
3		
4		
5		
6		
7		

Table 7.9

Compare the results between Table 7.3, 7.4 and Table 7.8, 7.9. Are they different?

_____

23. From Program 7.4, replace the BHI instruction with BLO instruction as follows:

BHI endif ➔ **BLO endif**

24. Repeat the step 19 and 20 again. Then observe and record the results in Table 7.10

The values of	**The results from BLO instruction**	
X	**R1**	**R2**
3		
4		
5		
6		
7		

Table 7.10

25. From Program 7.4, replace the BHI instruction with BLS instruction as follows:

BHI endif ➔ **BLS endif**

26. Repeat the step 19 and 20 again. Then observe and record the results in Table 7.11

The values of	**The results from BLS instruction**	
X	**R1**	**R2**
3		
4		
5		
6		
7		

Table 7.11

27. Create a new project, then type and add the following code to the project.

```
                AREA  Power_x_y,  CODE,  READONLY
                ENTRY
start           LDR   R1, =X          ; x = 3
                LDR   R2, =Y          ; y = 4
                CMP   R2, #0          ; if (y = 0)
                BNE   else1           ; {
                MOV   R0, #1          ;         result = 1
                B     endif1          ; }
else1                                 ; else {
                MOV   R0,  R1         ;         result = x
                CMP   R2,  #1         ;          if (y >1)
                BLS   endif2          ;         {
                SUBS  R2,  R2,  #1    ;                   y = y -1
do1                                   ;                   do {
                MUL   R0,  R1,  R0    ;                             result = result × x
                SUBS  R2,  R2,  #1    ;                              y = y -1
                BNE   do1             ;                   } while (y ≠0)
endif2                                ;         }
endif1                                ; }
stop            B     stop
                END
```

Program 7.5

28. Replace X and Y in the program above with the values as shown in Table 7.12.

29. Build and Run the program. Then observe and record the results in Table 7.12.

The values of		**The results from Program 7.5**
X	**Y**	**R0**
3	0	
3	2	
3	3	
5	3	
10	2	
20	1	
20	4	

Table 7.12

From the result shown in Table 7.12, what does the program do?

_____

_____

30. Modify the following program to replace the eight **ADD** instructions with iterative loop.

```
start
                LDR  R1, =0x12

                ADD  R0,  R0,  R1,  lsl  #1
                ADD  R0,  R0,  R1,  lsl  #2
                ADD  R0,  R0,  R1,  lsl  #3
                ADD  R0,  R0,  R1,  lsl  #4
                ADD  R0,  R0,  R1,  lsl  #5
                ADD  R0,  R0,  R1,  lsl  #6
                ADD  R0,  R0,  R1,  lsl  #7
                ADD  R0,  R0,  R1,  lsl  #8

stop            B       stop
```