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

Worksheet #5
Bitwise Operations

Objectives

- To understand the logical operators and their operations
- To comprehend the usage of bit manipulation instructions and their effects
- To understand the usage of shift and rotate instructions and their behavior

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

```
        AREA  PROG5_1,  CODE,  READONLY
        ENTRY
start
        LDR   R0, =X
        LDR   R1, =Y
        AND    R2,  R0, R1
loop    B       loop
        END
```

Program 5.1

2. Replace X and Y in the program above with the values as shown in Table 5.1.

3. Build and Run the program. Then observe and record the results in Table 5.1.

The values of		**The results from AND instruction**
X	**Y**	**R2**
0x57BD	**0x**0000	
0x57BD	**0x**FFFF	
0x12345678	**0x**000000FF	
0x12345678	**0x**0000FF00	
0x12345678	**0x**00FF0000	
0x12345678	**0x**FF000000	

Table 5.1

Write an instruction to **clear** bits 0, 2, 4, 6 of the value stored in R0 then store the result back to R0 (using AND instruction). **Hint** $1 \bullet x = x$ and $0 \bullet x = 0$.

_____

_____

4. From Program 5.1, replace the AND instruction with BIC instruction as follows:

AND R2, R0, R1 ➔ **BIC R2, R0, R1**

5. Repeat the step 2 and 3 again. Then observe and record the results in Table 5.2.

The values of		The results from BIC instruction
X	**Y**	**R2**
0x57BD	**0x**000F	
0x57BD	**0x**FFFF	
0x12345600	**0x**000000FF	
0x12345678	**0x**0000FF00	
0x12345678	**0x**00FF0000	
0x123456FF	**0x**FF000000	

Table 5.2

6. From Program 5.1, replace the AND instruction with ORR instruction as follows:

AND R2, R0, R1 ➔ **ORR R2, R0, R1**

7. Repeat the step 2 and 3 again. Then observe and record the results in Table 5.3.

The values of		The results from ORR instruction
X	**Y**	**R2**
0x57BD	**0x**0000	
0x57BD	**0x**FFFF	
0x12345600	**0x**000000FF	
0x12345678	**0x**0000FF00	
0x12345678	**0x**00FF0000	
0x123456FF	**0x**FF000000	

Table 5.3

Write an instruction to **set** bits 1, 3, 5, 7 of the value stored in R0 then store the result back to R0 (using ORR instruction). **Hint** 1 + x = 1 and 0 + x = x.

_____

_____

8. From Program 5.1, replace the AND instruction with EOR instruction as follows:

AND R2, R0, R1 ➔ **EOR R2, R0, R1**

9. Repeat the step 2 and 3 again. Then observe and record the results in Table 5.4.

The values of		The results from EOR instruction
X	**Y**	**R2**
0x7BD	**0x**123	
0x57A0	**0x**000F	
0x500A	**0x**0FF0	
0x0000	**0x**0000	
0x0000	**0x**FFFF	
0xFFFF	**0x**FF00	

Table 5.4

Write an instruction to **invert** bits 1, 3, 5, 7 of the value stored in R0 then store the result back to R0 (using EOR instruction). **Hint** $1 \oplus x = \overline{x}$ and $0 \oplus x = x$.

_____

_____

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

```
        AREA  PROG5_2,  CODE,  READONLY
        ENTRY
start
        LDR   R0, =0x10                 ----------①
        MOV   R1,  #X                   ----------②
        LSL     R2,  R0,  R1            ----------③
loop    B       loop
        END
```

Program 5.2

11. Replace X in the program above with the values as shown in Table 5.5.

12. Build and Run the program. Then observe and record the results in Table 5.5.

The values of	**The results from LSL instruction**
X	**R2**
0	
1	
2	
3	
4	
5	

Table 5.5

13. From Program 5.2, replace the instructions ① and ③ with:

LDR R0, =0x10 ➔ **LDR R0, =1024**
LSL R2, R0, R1 ➔ **LSR R2, R0, R1**.

14. Repeat the step 11 and 12 again. Then observe and record the results in Table 5.6.

The values of	**The results from LSR instruction**
X	**R2**
0	
1	
2	
3	
4	
5	

Table 5.6

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

```
        AREA  PROG5_3,  CODE,  READONLY
        ENTRY
start
        LDR   R0, =0x80123450
        MOV   R1,  #X
        ASR    R2,  R0,  R1
loop    B       loop
        END
```

Program 5.3

16. Replace X in the program above with the values as shown in Table 5.5.

17. Build and Run the program. Then observe and record the results in Table 5.5.

The values of	**The results from ASR instruction**	
X	**R2 in hex format**	**R2 in binary format**
0		
1		
2		
3		
4		
5		

Table 5.5

18. From Program 5.3, replace the ASR instruction with ROR instruction as follows:

ASR R2, R0, R1 ➔ **ROR R2, R0, R1**

19. Repeat the step 16 and 17 again. Then observe and record the results in Table 5.6.

The values of	**The results from ROR instruction**	
X	**R2 in hex format**	**R2 in binary format**
0		
1		
2		
3		
4		
5		

Table 5.6

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

```
        AREA  PROG5_4, CODE, READONLY
        ENTRY
start
        MOV   a1, #0xFF
        MOV   a2, #64
        XXX   a2, #2                        ----------②
        MOV    a3, #4
        ADD   a1, a1, a2, YYY a3            ----------④
        EOR    a2, a2, #0x65
        ORR   a1, a1, a2, ZZZ #8                    ----------⑥
loop    B      loop
        END
```

Program 5.4

16. Replace XXX, YYY and ZZZ in the line ②, ④ and ⑥ of Program 5.4 with the instructions shown in Table 5.7.

17. Build and Run the program. Then observe and record the results in Table 5.7.

Replace…with			**Results in**		
XXX	**YYY**	**ZZZ**	**R0**	**R1**	**R2**
LSL	LSL	LSL			
LSR	LSR	LSR			
ASR	ASR	ASR			
ROR	ROR	ROR			

Table 5.7

18. Complete the following program to make data stored in registers as follow.
R2 = 0x10305070, R3 =0xE2C4A688, R4 = 0x10300000

```
        AREA  PROG5_5, CODE, READONLY
        ENTRY
start
        LDR   R0, =0x12345678
        LDR   R1, =0xF0F0F0F0

        _________________ R2, R0, R1

        _________________ R3, R0, R1

        AND    R4, R0, R1, _________________

loop    B   loop
        END
```

Program 5.5

19. Suppose that R0 stores an arbitrary 32-bit value before running the program. Write an assembly program that separates the value in R0 into 4 groups of byte. Then do the addition all these 4 groups of byte together and store the result of addition in R7.

Ex. R0 = 0x12345678

4 groups of byte = [12] [34] [56] [78]

The result is stored in R7 = [12] + [34] + [56] + [78]