Experiment 8

Code:

```
DATA SEGMENT
Q DB 8H
M DB 5H
RESULT DW ?
 DB 100 DUP(00)
DATA ENDS



CODE SEGMENT
ASSUME CS: CODE, DS: DATA

START:  MOV AX, DATA
    MOV DS, AX

    MOV AX, 0000H
    MOV BX, 0000H
    MOV AL, Q
    MOV BL, M
    MOV CL, 8
    CLC
BACK:   MOV DX, 00H
    MOV DX, AX
    RCL DX, 01
    AND DX, 03H

    CMP DX, 01H
```

```
    JE Q01
    CMP DX, 02H
    JE Q10
NEXT:  SAR AX, 01
    LOOP BACK
    JMP EXIT

Q10:   SUB AH, BL
    JMP NEXT
Q01:   ADD AH, BL
    JMP NEXT

EXIT:  MOV RESULT, AX

    INT 3H
CODE ENDS
END START
```

Explanation:

DATA SEGMENT

This section defines the data variables that will be used in the program.

1. Q DB 8H
    - Q is defined as a byte with a value of 8H (which is 8 in hexadecimal or 8 in decimal).
2. M DB 5H
    - M is defined as a byte with a value of 5H (which is 5 in hexadecimal or 5 in decimal).
3. RESULT DW ?
    - RESULT is defined as a word (16-bit value). The question mark (?) indicates that its value is uninitialized.
4. DB 100 DUP(00)

- This line defines a sequence of 100 bytes, each initialized to 00. It's likely used as a buffer or padding.

5. DATA ENDS
    - Marks the end of the data segment.

CODE SEGMENT

This section defines the program's instructions.

6. ASSUME CS: CODE, DS: DATA
    - This tells the assembler that the code segment (CS) is the segment named CODE, and the data segment (DS) is the segment named DATA.
7. START: MOV AX, DATA
    - START is the label where the program begins execution. The instruction moves the address of the data segment into the AX register.
8. MOV DS, AX
    - Moves the value of AX into the data segment register (DS).
9. MOV AX, 0000H
    - Initializes the AX register to 0000H (0 in hexadecimal).
10. MOV BX, 0000H

- Initializes the BX register to 0000H.

11. MOV AL, Q

- Loads the lower byte of AX (register AL) with the value of Q (which is 8H).

12. MOV BL, M

- Loads the lower byte of BX with the value of M (which is 5H).

13. MOV CL, 8

- Loads the CL register with the value 8, setting up a loop counter.

14. CLC

- Clears the carry flag (CLC stands for "Clear Carry Flag").

BACK LOOP:

This part of the program defines the loop that will iterate 8 times.

15. BACK: MOV DX, 00H

- The label BACK marks the start of the loop. The instruction sets register DX to 00H.

16. MOV DX, AX

- Copies the value in AX into DX.

17. RCL DX, 01

- Performs a Rotate Through Carry Left (RCL) operation on DX by 1 bit.

18. AND DX, 03H

- Performs a bitwise AND operation between DX and 03H (which is 00000011 in binary).

19. CMP DX, 01H

- Compares DX with 01H.

20. JE Q01

- If DX == 01H, jump to the label Q01.

21. CMP DX, 02H

- Compares DX with 02H.

22. JE Q10

- If DX == 02H, jump to the label Q10.

NEXT:

If neither jump to Q01 nor Q10 occurs, the program moves to the NEXT section to continue the loop.

23. NEXT: SAR AX, 01

- Shift Arithmetic Right (SAR) shifts the bits of AX to the right by 1 bit.

24. LOOP BACK

- The LOOP instruction decrements CX by 1 and, if CX is not zero, jumps back to the BACK label.

Q10:

This section executes when the least significant bits are 10 after the RCL operation.

25. Q10: SUB AH, BL

- Subtracts the value of M (which is 5H) from the high byte of AX (AH).

26. JMP NEXT

- Jumps to the NEXT label to continue the loop.

Q01:

This section executes when the least significant bits are 01 after the RCL operation.

27. Q01: ADD AH, BL

- Adds the value of M (which is 5H) to the high byte of AX (AH).

28. JMP NEXT

- Jumps to the NEXT label to continue the loop.

EXIT:

This section is where the program ends, storing the result and halting execution.

29. EXIT: MOV RESULT, AX

- The value in AX is moved to RESULT. This stores the result of all operations.

30. INT 3H

- The INT 3H instruction triggers a breakpoint interrupt, halting the program.

31. CODE ENDS

- Marks the end of the code segment.

32. END START

- Marks the end of the program, and specifies that the execution starts from the START label.