Experiment 4

Code:

```
ASSUME CS:CODE


CODE SEGMENT
START:MOV BX, 1234H
MOV CX, 7698H


MOV AL, BL      ; AL = 34H
ADD AL, CL      ; AL = 34H + 98H = CCH
DAA             ; AL becomes 32H, CF = 1
MOV DL, AL


MOV AL, BH      ; AL = 12H
ADC AL, CH      ; AL = 12H + 76H + 1 = 89H
DAA             ; AL stays 89H
MOV DH, AL      ; DH = 89H
INT 21H
CODE ENDS
END START
```

Explanation:

1. ASSUME CS:CODE
   - Assembler directive (not executed). Tells the assembler that the CODE segment will be loaded into the CS register.
2. CODE SEGMENT
   - Start of the code segment.
3. START:

- Label marking the entry point.

4. MOV BX, 1234H
   - Move immediate 0x1234 into BX.
   - Result: BX = 1234H → BH = 12H, BL = 34H.
5. MOV CX, 7698H
   - Move immediate 0x7698 into CX.
   - Result: CX = 7698H → CH = 76H, CL = 98H.

(At this point we can view BX and CX as two packed BCD numbers: 12|34 and 76|98 — i.e., decimal 1234 and 7698.)

6. MOV AL, BL ; AL = 34H
   - Copy low byte of BX into AL.
   - Result: AL = 34H (decimal 52). MOV does not change flags.
7. ADD AL, CL ; AL = 34H + 98H = CCH
   - Add CL (98H) to AL (34H): 0x34 + 0x98 = 0xCC.
   - Result: AL = CCH. Flags set by ADD:
     - CF = 0 (no carry beyond 8 bits here),
     - ZF = 0 (AL ≠ 0),
     - SF = 1 (0xCC has MSB=1),
     - AF = 0 (no nibble carry from bit3 in raw add),
     - OF depends on signed overflow rules (not relevant for BCD here).
8. DAA ; AL becomes 32H, CF = 1
   - DAA = Decimal Adjust AL after Addition: used to correct the result of adding two packed BCD bytes.
   - How it works here:
     - After ADD AL = CC (nibbles: high=C (12), low=C (12)). Low nibble > 9 → add 06 → CC + 06 = D2.
     - Result D2 > 99 → add 60 → D2 + 60 = 132 (0x132). Only low 8 bits remain in AL → AL = 32H, and a carry out sets CF = 1. AF will also be adjusted by DAA as needed.
   - Semantically: interpreted as decimal 34 + 98 = 132 → low byte 32 and carry 1 into the next decimal digit.

  - After DAA: AL = 32H (represents decimal 32), CF = 1.

9. MOV DL, AL
   - Store the adjusted low-order BCD byte into DL.
   - Result: DL = 32H. (This is the lower two decimal digits of the BCD sum.)

10. MOV AL, BH ; AL = 12H
    - Copy high byte of BX into AL.
    - Result: AL = 12H.

11. ADC AL, CH ; AL = 12H + 76H + CF = 89H
    - ADC = add with carry. It adds CH and the previous CF (which was set to 1 by the earlier DAA).
    - Calculation: 0x12 + 0x76 + 1 = 0x89.
    - Result: AL = 89H. Flags after ADC:
      - CF = 0 (no carry out from this 8-bit addition),
      - ZF = 0 (AL ≠ 0),
      - SF = 1 (0x89 MSB=1),
      - AF as affected by the nibble add.
    - Conceptually: this is adding the high BCD bytes plus the carry from the low-byte BCD addition: 12 + 76 + 1 = 89 (decimal).

12. DAA ; AL stays 89H
    - AL = 89H is already a valid packed BCD (nibbles 8 and 9 are ≤ 9), so DAA does no adjustment.
    - AL remains 89H. CF remains 0 (no extra carry).

13. MOV DH, AL ; DH = 89H
    - Store the adjusted high-order BCD byte into DH.
    - Result: DH = 89H. (DH:DL = 89H:32H — packed BCD for decimal 8932.)

14. INT 21H
    - DOS interrupt. Behavior depends on value in AH (the function number). In this code AH is not set, so the effect is undefined/depends on whatever AH contained.
    - Typical convention to terminate a program would be to set AH = 4Ch and AL = return_code then INT 21h. Here that was not done, so this INT 21h is incomplete as a proper program exit or DOS call.

15. CODE ENDS

   - End of code segment (assembler directive).

16. END START

   - Marks the end of the source file and that START is the program entry point.

Final state and high-level meaning

- Input BCD values: BX = 1234H (decimal 1234), CX = 7698H (decimal 7698).
- The code performs packed-BCD addition of these 16-bit BCD numbers:
  - Lower byte: 34 + 98 = 132 → stored as 32 in DL, carry 1.
  - Higher byte: 12 + 76 + carry(1) = 89 → stored in DH.
- Final BCD result = DH:DL = 89H:32H which represents decimal 8932 — equals 1234 + 7698.
- Key registers after execution (relevant ones):
  - BX = 1234H, CX = 7698H (unchanged),
  - DL = 32H, DH = 89H,
  - AL = 89H (last value moved into DH),
  - CF = 0 (after the final DAA/ADC sequence),
  - AH is undefined (so INT 21h behaviour is undefined).