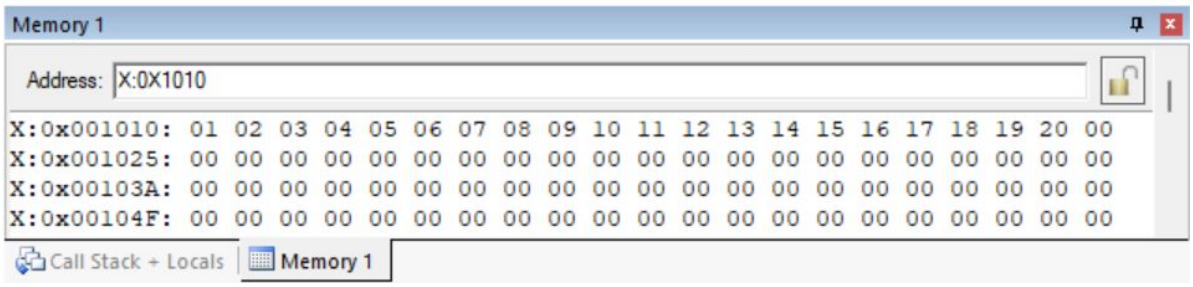
Experiment 3

Memory Array Handling and Code conversion

Exercise 1

Input

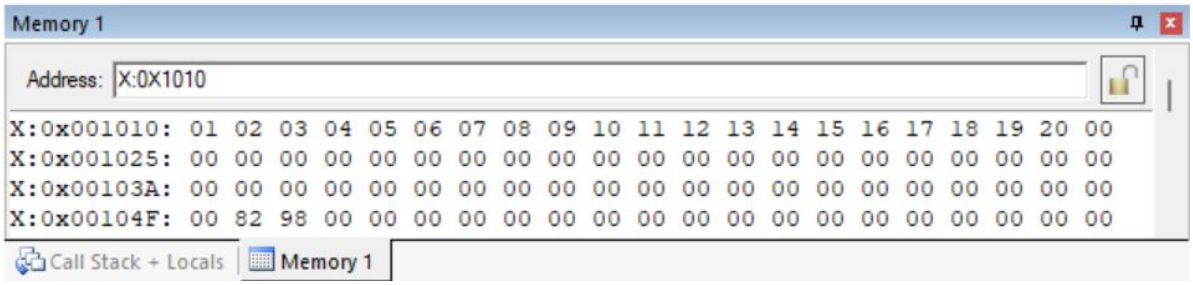


Code

```
; Find the sum of ten 16 -bit hexadecimal numbers available in memory
; starting from location XX10h. store the result in XX50h onwards.

ORG 0000H               ; ORIGINATE
AJMP START              ; JUMP TO THE LABEL START

START:
    MOV DPTR, #1010H    ; STARTING LOCATION OF THE MEMORY
    MOV R0, #00H        ; LOWER SUM REGISTER; CLR (R0) DOESN'T WORK;
    MOV R1, #00H        ; HIGHER SUM REGISTER; CLR (R1) DOESN'T WORK;
    MOV R2, #00H        ; CARRY REGISTER; CLR (R2) DOESN'T WORK;
    MOV R3, #10         ; NUMBER OF ELEMENTS, DECIMAL 10

ITER:
    MOVX A, @DPTR       ; GET THE LOWER BYTE OF THE DATA
    MOV 0F0H, A         ; DUPLICATE IT TO REG (B)
    MOV A, R0           ; GET THE CURRENT SUM, LOWER BYTE.
    ADD A, B            ; ADD THE LOWER BYTE TO THE SUM, WITHOUT CARRY.
    MOV R0, A           ; SAVE THE LOWER BYTE OF SUM IN (R0)

    INC DPTR            ; GO TO NEXT LOCATION
    MOVX A, @DPTR       ; GET THE HIGHER BYTE OF THE DATA
    MOV 0F0H, A         ; DUPLICATE IT TO REG (B)
    MOV A, R1           ; GET THE CURRENT SUM, HIGHER BYTE.
    ADDC A, B           ; ADD THE HIGHER BYTE, ALONG WITH THE CARRY.
    MOV R1, A           ; SAVE THE HIGHER BYTE OF SUM IN (R1)

    MOV A, R2           ; GET THE CURRENT SUM, CARRY BYTE.
    ADDC A, #00H        ; INCREMENT CARRY REGISTER WITH THE CARRY.

    INC DPTR            ; GO TO NEXT LOCATION
    DJNZ R3, ITER       ; REPEAT TILL END OF ARRAY
```

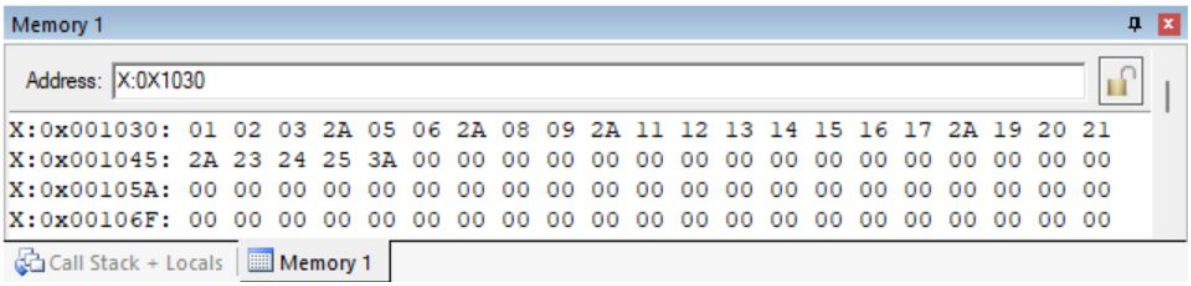
```
STORE:
    ; STORE THE FINAL SUM IN THE DESIRED LOCATION
    MOV DPTR, #1050H     ; LOWER BYTE OF SUM
    MOV A, R0
    MOVX @DPTR, A

    INC DPTR             ; HIGHER BYTE OF SUM
    MOV A, R1
    MOVX @DPTR, A

    INC DPTR             ; CARRY BYTE OF SUM
    MOV A, R2
    MOVX @DPTR, A

HERE:
    SJMP HERE            ; LOGICAL END
    END
```

Output



Exercise 2

Input

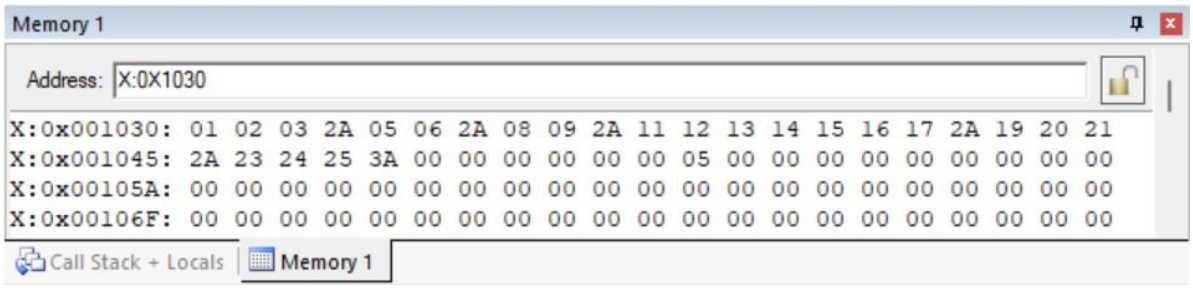


Code

```
; Find the number of occurrences of data ‘2Ah’ in a memory array.
; The last element of the array is ‘3Ah’. The array begins at XX30h.
; Store the result in memory location XX80h.

ORG 0000H                ; ORIGINATE
AJMP START               ; JUMP TO THE LABEL START

START:
    MOV DPTR, #1030H     ; THE START LOCATION OF THE ARRAY
    MOV R0, #2AH         ; THE VALUE TO BE COMPARED FOR EQUALITY
```

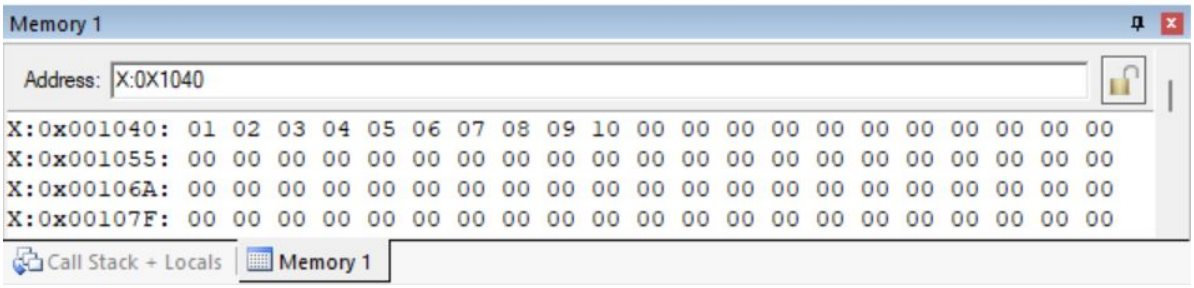
```
    CLR C                ; CLEAR THE CARRY
    MOV R1, #00H         ; COUNT REGISTER; CLR (R1) DOESN'T WORK;

REPEAT:
    MOVX A, @DPTR        ; GET THE DATA
    SUBB A, R0           ; PERFORM A SUBTRACTION TO CHECK FOR EQUALITY
    INC DPTR             ; INCREMENT TO THE NEXT ARRAY POSITION

    ; IF THE DIFFERENCE IS NOT ZERO
    JNZ CHECK

    ; IF THE DIFFERENCE IS ZERO, UPDATE COUNT AND PROCEED TO CHECK
    INC R1

CHECK:
    ; CHECK WHETHER TO STOP
    SUBB A, #10H         ; SUBTRACT 10H TO CHECK FOR STOP CONDITION

    ; REPEAT THE PROCESS IF NOT REACHED THE STOP CONDITION
    JNZ REPEAT

STORE:
    ; STOP AND STORE THE FINAL COUNT IN THE DESIRED LOCATION
    MOV A, R1
    MOV DPTR, #1080H
    MOVX @DPTR, A

HERE:
    SJMP HERE            ; LOGICAL END
    END
```

Output



Exercise 3

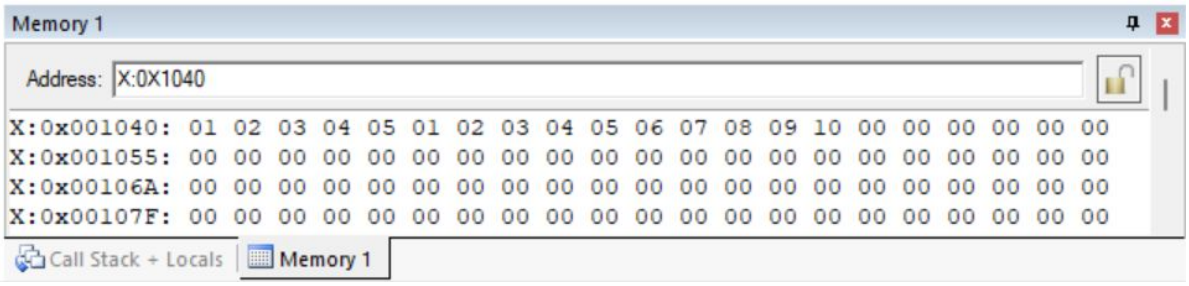
Input



Code

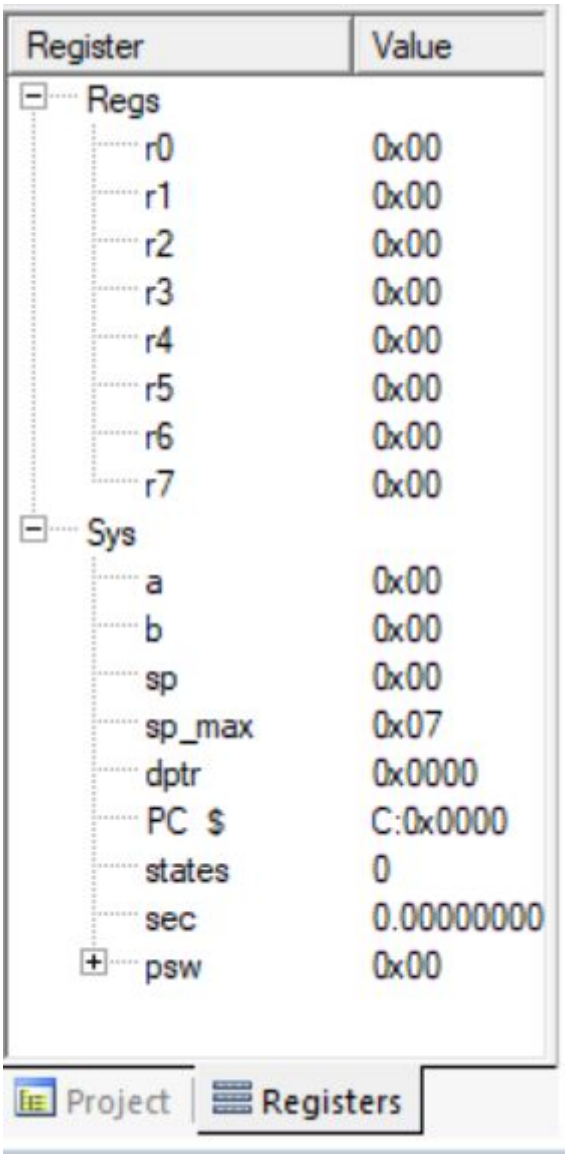
```
; Transfer ten elements of an array starting at location XX40h
; in external data memory to a location XX45 in the same memory

ORG 0000H                ; ORIGINATE
AJMP START               ; JUMP TO THE LABEL START

START:
    MOV DPTR, #1049H     ; GO TO LAST ELEMENT OF THE ARRAY
    MOV R0, #10          ; NUMBER OF ELEMENTS, DECIMAL 10
    MOV R1, #05H         ; NUMBER OF ELEMENTS, DECIMAL 5, I.E. (XX49 - XX45 + 1)

REPEAT:
    MOV A, R1            ; GET DISPLACEMENT
    MOV R2, A            ; COPY TO (R2)

    MOVX A, @DPTR        ; GET THE DATA TO ACCUMULATOR

; INCREMENT DPTR BY (R1)
INCREMENT:
    INC DPTR
    DJNZ R2, INCREMENT

    MOVX @DPTR, A        ; COPY DATA TO DESTINATION

    MOV A, R1            ; GET DISPLACEMENT AGAIN SINCE R2 IS DESTROYED
    MOV R2, A            ; COPY TO (R2)
    INC R2               ; SO AS TO POINT TO PREVIOUS LOCATION IN ORIGINAL ARRAY

; DECREMENT DPTR BY (R1)+1; DEC DPTR IS NOT AVAILABLE
DECREMENT:
    CLR C                ; CLEAR CARRY
    MOV A, DPL           ; GET DPTR LOW
    SUBB A, #01H         ; DECREMENT DPL
    MOV DPL, A           ; RESTORE DPL

    MOV A, DPH           ; GET DPTR HIGH
    SUBB A, #00H         ; SUBTRACT CARRY [IF EXISTS] FROM (DPH)
    MOV DPH, A           ; RESTORE DPH

    DJNZ R2, DECREMENT   ; REPEAT UNTIL (R2) IS ZERO

    DJNZ R0, REPEAT      ; LOOP UNTIL ARRAY LENGTH IS COVERED

HERE:
    SJMP HERE            ; LOGICAL END
    END
```

Output



Exercise 4

Input



Code

```
; Convert a 2-digit hexadecimal number into BCD number.

; LIMITATIONS: ACCEPTS INPUT 2-DIGIT HEXADECIMAL IN DECIMAL RANGE

ORG 0000H                ; ORIGINATE
AJMP START               ; JUMP TO THE LABEL START
```

```
START:
    MOV R0, #92H        ; INPUT 2-DIGIT HEXADECIMAL NUMBER, IN DECIMAL RANGE
    MOV A, R0           ; COPY TO THE ACCUMULATOR

    ANL A, #0FH         ; MASK THE UPPER NIBBLE. RETURNS LOWER NIBBLE
    MOV R1, A           ; STORE LOWER NIBBLE IN (R1)

    MOV A, R0           ; COPY TO THE ACCUMULATOR AGAIN SINCE INPUT IS DESTROYED
    ANL A, #0F0H        ; MASK THE LOWER NIBBLE. RETURNS UPPER NIBBLE
    SWAP A              ; SWAP THE CURRENT LOWER AND UPPER NIBBLE IN ACCUMULATOR
    MOV R2, A           ; STORE UPPER NIBBLE IN (R2)

HERE:
    SJMP HERE           ; LOGICAL END
    END
```

Output