Course Name:	**Microprocessors and Microcontrollers Laboratory**	**Semester:**	**IV**
Date of Performance:	**08 / 04 / 2024**	**Batch No.:**	**A - 2**
Faculty Name:	**Kirti Sawlani**	**Roll No.:**	**16014022050**
Faculty Sign & Date:		**Grade / Marks:**	**___ / 25**

Experiment No.: 5
Title: Data Block Transfer within Internal RAM

Aim and Objective of the Experiment:

Aim:

1. **Write an 8051 based ALP to copy a given number of bytes from memory location 50H to 60H.**
2. **Write an 8051 based ALP to exchange number of bytes from memory location 50H to 60H.**

Objectives:

Study of –

1. Assembler directives.
2. Direct and indirect addressing modes.
3. MOV instruction.
4. RAM organization of 8051.

COs to be achieved:

CO3: Understand the internal design of 8051 microcontrollers along with its features.

Theory:

8051 microcontroller is the CISC based Harvard architecture designed by Intel in 1981. It is an 8-bit microcontroller. It is built with 40 pins DIP (dual inline package), 4kb of ROM storage and 128 bytes of RAM storage, 2 16-bit timers. It consists of are four parallel 8-bit ports, which are programmable as well as addressable as per the requirement. An on-chip crystal oscillator is integrated in the microcontroller having crystal frequency of 12 MHz.

8051 Programming in Assembly Language is the assembly language is a fully hardware related programming language. The embedded designers must have sufficient knowledge on hardware of particular processor or controllers before writing the program. The assembly language is developed by mnemonics; therefore, users cannot understand it easily to modify the program. Microcontrollers or processors can understand only binary language in the form of '0s or 1s'; An assembler converts the assembly language to binary language, and then stores it in the microcontroller memory to perform the specific task.

The Elements of an Assembly Language Programming:

Assembler Directives are the instructions used by the assembler at the time of assembling a source program. These are the instructions provided to the assembler, not the processor as the processor has nothing to do with these instructions. org, equ, db, end are the assembler directives.

Instruction Set

Stepwise-Procedure:

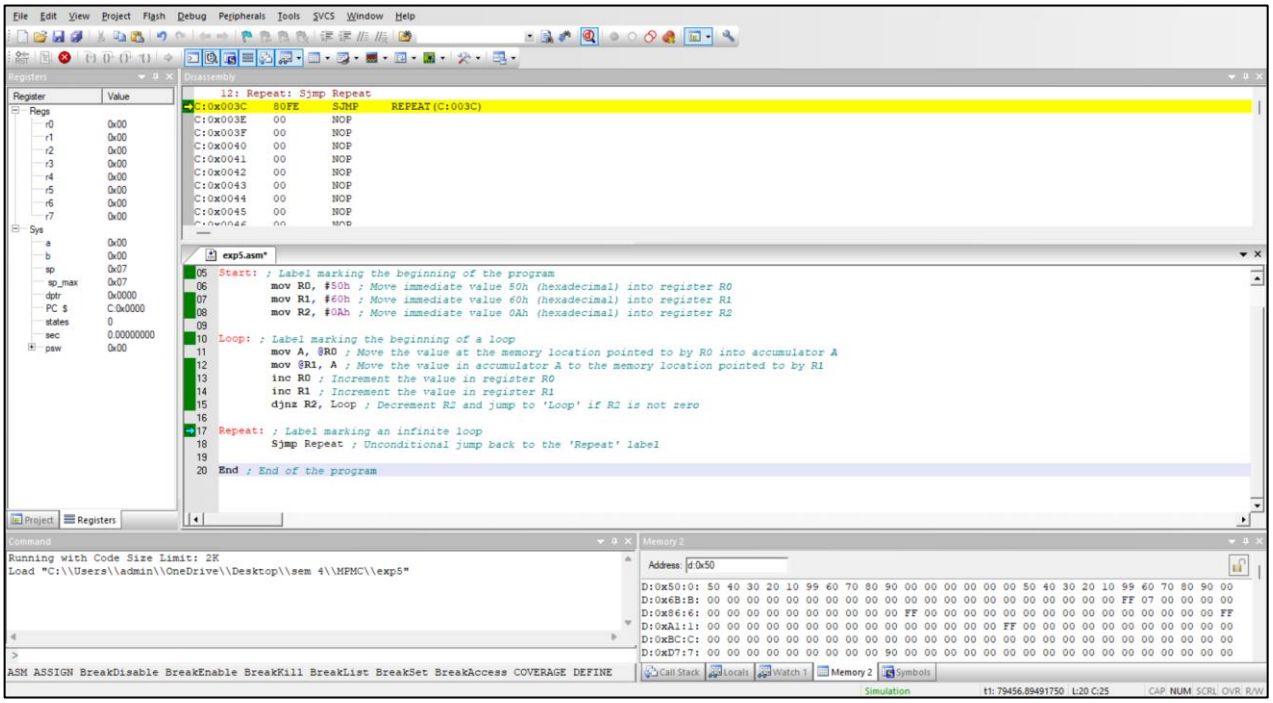
1. Write an assembly program to achieve the aim.
2. Build the program using Keil microvision-3 IDE.
3. Run the program on in-built simulator and observe the contents of the internal RAM.

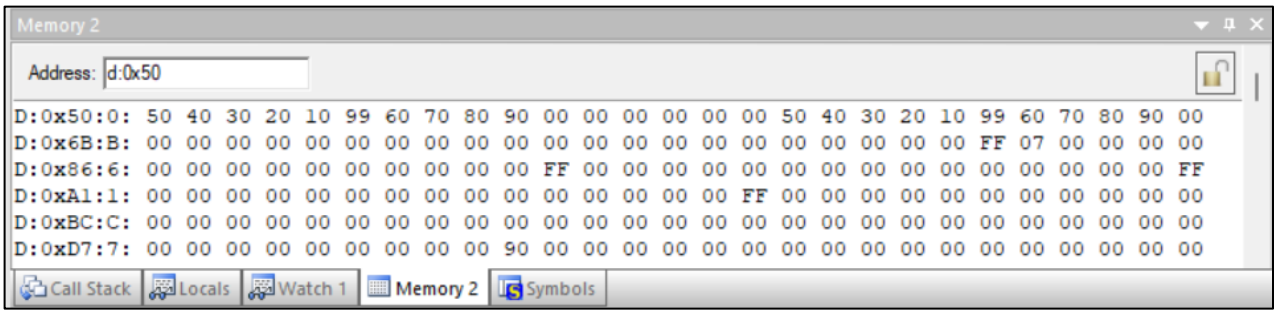
Assembly Language Program:

```
A51 MACRO ASSEMBLER  EXP5                                  04/08/2024 14:23:20 PAGE     1

MACRO ASSEMBLER A51 V8.02
OBJECT MODULE PLACED IN exp5.OBJ
ASSEMBLER INVOKED BY: C:\Keil\C51\BIN\A51.EXE exp5.asm SET(SMALL) DEBUG EP

LOC  OBJ          LINE     SOURCE

0000              1     org 0h                         ; Set the origin of the program counter to address 0
0000 802E         2     sjmp start                     ; Unconditional jump to the 'start' label
```

```
0030                3     org 30h                         ; Set the origin of the program counter to address 30h

0030 7850           4     start:          mov R0,#50h  ; Move immediate value 50h (hexadecimal) into register R0
0032 7960           5                     mov R1,#60h  ; Move immediate value 60h (hexadecimal) into register R1
0034 7A0A           6                     mov R2,#0Ah  ; Move immediate value 0Ah (hexadecimal) into register R2
0036 E6             7     Loop:           mov A,@R0    ; Move the value at the memory location pointed to by R0 into
accumulator A
0037 F7             8                     mov @R1,A    ; Move the value in accumulator A to the memory location pointed to
by R1
0038 08             9                     inc R0       ; Increment the value in register R0
0039 09             10                    inc R1       ; Increment the value in register R1
003A DAFA           11                    djnz R2,Loop ; Decrement R2 and jump to 'Loop' if R2 is not zero

003C 80FE           12    Repeat: Sjmp Repeat           ; Unconditional jump back to the 'Repeat' label
                    13    End                           ; End of the program
A51 MACRO ASSEMBLER  EXP5                                        04/08/2024 14:23:20 PAGE     2

SYMBOL TABLE LISTING
------ ----- -------

N A M E             T Y P E  V A L U E   ATTRIBUTES

LOOP . . . . . .  C ADDR   0036H   A
REPEAT . . . . . .  C ADDR   003CH   A
START. . . . . .  C ADDR   0030H   A

REGISTER BANK(S) USED: 0

ASSEMBLY COMPLETE.  0 WARNING(S), 0 ERROR(S)
```

Output Screenshots:



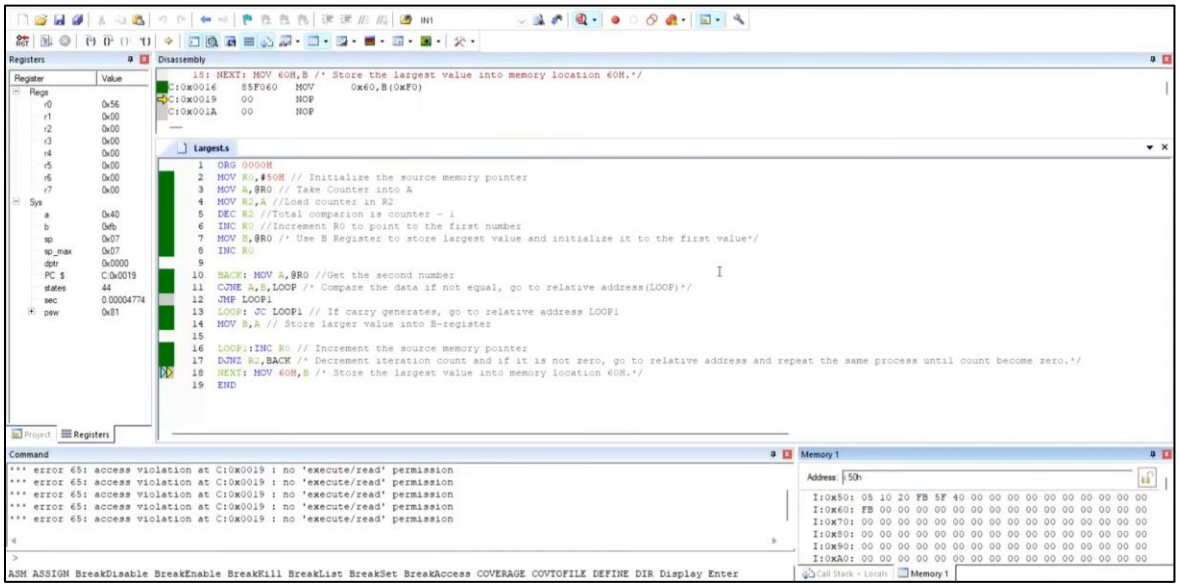


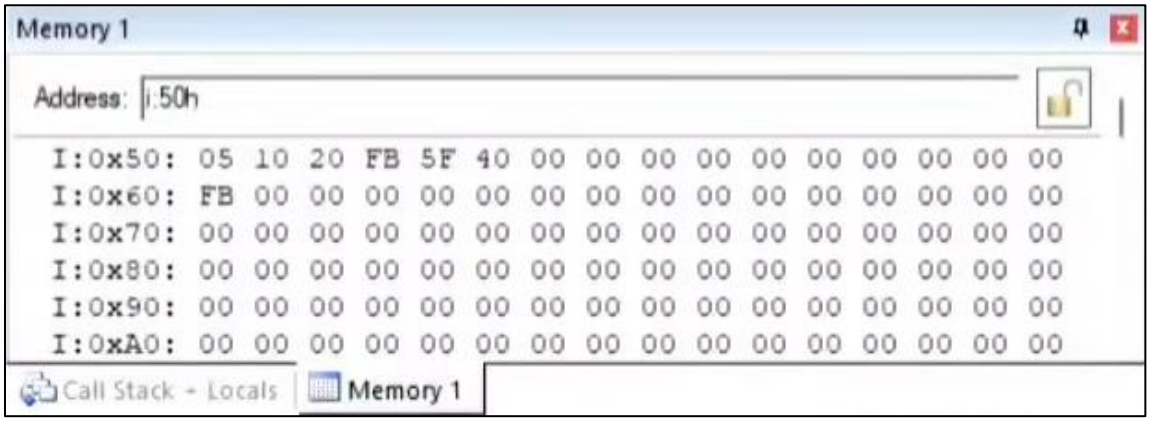
Post Lab Subjective/Objective type Questions:

1. **WAP to find largest number from numbers stored in RAM memory.**

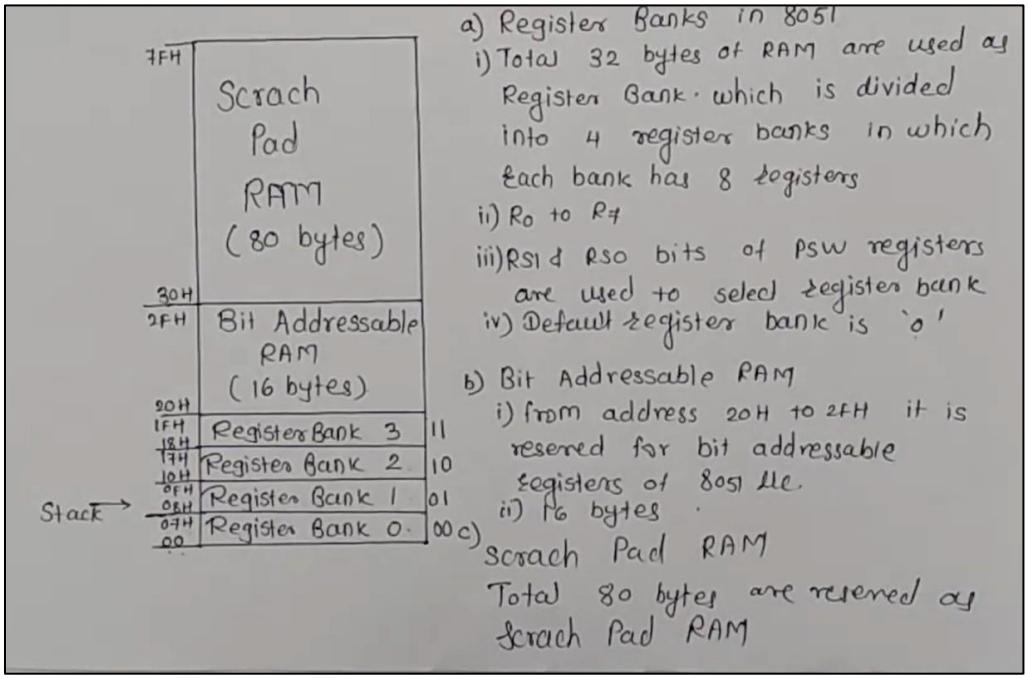
```
ORG 0H      ; Start of the program

MOV R0, #30H  ; Initialize R0 with the starting address of the memory block
MOV R1, #00H  ; Initialize R1 with the count of numbers
MOV A, #00H   ; Initialize accumulator with 0 (to store largest number)
MOV B, #00H   ; Initialize B register with 0 (temporarily stores a number for comparison)
```

```
LOOP:
   MOVX A, @R0  ; Move data from the memory location pointed by R0 to accumulator
   CJNE A, #00H, COMPARE  ; Compare the data with 0, if not 0 then go to COMPARE
   SJMP END_LOOP        ; If data is 0, end of the list
COMPARE:
   CJNE A, B, CHECK_NEXT  ; Compare the data with the current largest number
   SJMP CHECK_NEXT        ; If data is less than or equal to current largest, check next number
   MOV B, A             ; If data is greater than current largest, update B with this data
CHECK_NEXT:
   INC R0          ; Move to the next memory location
   INC R1          ; Increment the count
   DJNZ R1, LOOP    ; Decrement R1 and jump to LOOP if it's not zero
END_LOOP:
   MOV A, B       ; Move the largest number to accumulator
   ; Here you can perform any further operations with the largest number
   ; For example, you can store it in a specific location in RAM or use it as required

END
```





2. Draw and explain internal RAM and ROM of 8051.



Conclusion:

In conclusion, the experiment successfully demonstrated the efficient Data Block Transfer within the Internal RAM of the 8051 microprocessors. This method proves to be effective in optimizing memory utilization and enhancing data handling capabilities

Signature of faculty in-charge with Date: