Experiment No. 2: Memory Interfacing with 8085 Microprocessor

1. Aim

To understand the fundamental principles of memory interfacing with the 8085 microprocessor, including memory mapping, address decoding, and to perform read/write operations with RAM and read operations with ROM.

2. Objectives

Upon completion of this experiment, the student will be able to:

- **Design a memory map for a given RAM/ROM configuration.**
- **Understand the role of address bus, data bus, and control signals in memory access.**
- **Identify and connect appropriate address decoding logic for memory chips.**
- **Write and execute 8085 assembly language programs to store and retrieve data from specific memory locations.**
- **Verify memory operations using the 8085 trainer kit's debugging features.**

3. Theory

Memory is crucial for storing programs and data in any microprocessor system. The 8085 microprocessor has a 16-bit address bus (A0-A15), allowing it to address 216 (65,536 or 64 KB) unique memory locations. Each location stores 8 bits of data, accessed via the 8-bit bidirectional data bus (D0-D7). Control signals like overlineRD (Read), overlineWR (Write), and IO/overlineM (I/O or Memory selection) orchestrate data transfer.

Memory can be broadly classified into:

- **Read-Only Memory (ROM): Non-volatile, used for permanent storage of boot programs and firmware. Data can only be read.**
- **Random Access Memory (RAM): Volatile, used for temporary storage of active programs and data. Data can be both read from and written to. SRAM (Static RAM) is often used in trainer kits due to simpler interfacing without refresh cycles.**

Memory Map: A memory map is a graphical or tabular representation of how the 8085's entire 64 KB address space is divided and allocated among various memory chips (RAM, ROM) and I/O devices. It ensures non-overlapping address ranges for each component.

To calculate the number of address lines required for a memory chip of size N bytes, the formula is:

Number of Address Lines = log_2(N)

For example, a 2KB (2048 bytes) memory chip requires log_2(2048)=11 address lines (A0-A10).

Address Decoding: Since memory chips typically have fewer address lines than the 8085's 16-bit address bus, the higher-order address lines must be decoded to generate a unique Chip Select (overlineCS or CE) signal for each memory chip. This prevents multiple chips from responding to the same address, ensuring proper memory access. Decoding can be achieved using logic gates (AND, OR, NAND, NOR, Inverters) or dedicated decoder ICs like the 74LS138 (a 3-to-8 line decoder).

Memory Read/Write Cycles:

- **Memory Read: The 8085 places the 16-bit address on the address bus, sets IO/overlineM low, and activates overlineRD (low). The selected memory chip places its data onto the data bus, which the 8085 then reads.**
- **Memory Write: The 8085 places the 16-bit address on the address bus, sets IO/overlineM low, places the 8-bit data on the data bus, and activates overlineWR (low). The selected memory chip then latches the data from the data bus into the addressed location.**

4. Materials Required

- **8085 Microprocessor Trainer Kit**
- **Compatible RAM chip (e.g., 6264 - 8KB SRAM or similar)**
- **Compatible ROM chip (e.g., 2716 - 2KB EPROM or similar, or built-in trainer kit ROM)**
- **Logic gates (e.g., 74LS00, 74LS04, 74LS08, 74LS32) or a Decoder IC (e.g., 74LS138) for address decoding (if external interfacing is required beyond the trainer kit's built-in memory).**
- **Connecting Wires/Jumper Cables**
- **Power Supply (usually integrated with the trainer kit)**

5. Procedure

Part A: Memory Map Design and Interfacing Schematic

1. **Given Memory Configuration:**
   - **ROM: 2 KB**
   - **RAM: 4 KB**
2. **Calculate Address Lines Required for Each Chip:**
   - **ROM (2KB): 2times1024=2048 bytes. Requires log_2(2048)=11 address lines (A0-A10).**
   - **RAM (4KB): 4times1024=4096 bytes. Requires log_2(4096)=12 address lines (A0-A11).**
3. **Assign Memory Addresses and Create Memory Map:**

- ROM: Assign starting address 0000H.
    - Ending Address = 0000H+2048−1=07FFH.
    - Address Range: 0000H−07FFH.
- RAM: Assign starting address 2000H (leaving a gap for other potential devices/expansion).
    - Ending Address = 2000H+4096−1=2FFFH.
    - Address Range: 2000H−2FFFH.

4. Complete Memory Map Table:

Memory Device	Size	Starting Address	Ending Address	Address Lines Used by Chip	Decoding Address Lines (for Chip Select)
ROM	2KB	0000H	07FFH	A0-A10	A11, A12, A13, A14, A15 (all must be 0)
RAM	4KB	2000H	2FFFH	A0-A11	A12, A13, A14, A15 (A12=1, others=0)

5. Design Address Decoding Logic and Interfacing Schematic:
    - Common Connections:
        - 8085 A0-A10 to ROM A0-A10.
        - 8085 A0-A11 to RAM A0-A11.
        - 8085 D0-D7 to ROM D0-D7 and RAM D0-D7.
        - 8085 overlineRD to ROM overlineOE and RAM overlineOE.
        - 8085 overlineWR to RAM overlineWE (ROM does not have overlineWE).
    - Chip Select (overlineCS) Generation:
        - For ROM (0000H−07FFH): Requires A15=0, A14=0, A13=0, A12=0, A11=0.
            - Decoding logic: Connect A11, A12, A13, A14, A15 to a 5-input NOR gate. The output of the NOR gate connects to the ROM's overlineCS.
            - *Alternative (more common for simple blocks):* If A11 is the primary differentiator for the lowest block, overlineA_11 can directly enable it, assuming higher bits are implicitly zero for this range. For absolute decoding, use inverters on A11-A15 feeding an AND gate, and that output to CS.
        - For RAM (2000H−2FFFH): Requires A15=0, A14=0, A13=0, A12=1.

        - **Decoding logic: Connect A15, A14, A13 (inverted) and A12 (direct) to a 4-input AND gate. The output of the AND gate connects to the RAM's overlineCS.**
    - **Master Memory Enable: The IO/overlineM signal (low for memory operations) should be incorporated into the overall chip select logic for each device (e.g., ANDed with the address decode output before connecting to overlineCS).**
    - **Conceptual Interfacing Diagram (similar to Theory section, to be drawn clearly in the practical file):**

```
          8085 Microprocessor
           +-----------------+
           |                 |
 A0-A15 ------| Address Bus     |
           |                 |
 D0-D7 -------| Data Bus       |
           |                 |
 IO/M --------| Control Signals |
 RD ----------|                 |
 WR ----------|                 |
           +-----------------+
                 |
                 | Address Bus, Data Bus, Control Signals
                 V

      +--------------------------------+
      |                                |
      |      Address Decoding Logic    |
      |      (Gates/Decoder IC)        |
      |                                |
```

    -
    -

A11-A15 & IO/M--| Input Address Lines & Control | --- CS_ROM, CS_RAM

| |

+--------------------------------+

|

+------------------+------------------+

| |

V V

```
+-------------+                    +-------------+
|    ROM      |                    |    RAM      |
| (e.g., 2KB) |                     | (e.g., 4KB) |
+-------------+                    +-------------+
 A0-A10 --- A0-A10                      A0-A11 --- A0-A11
 D0-D7  --- D0-D7                     D0-D7  --- D0-D7
 CS_ROM --- CS                          CS_RAM --- CS
 RD     --- OE                        RD     --- OE
                                WR     --- WE
   ```
```

**Part B: Assembly Language Programs and Execution**

**The following programs will be entered and executed on the 8085 trainer kit to demonstrate memory operations.**

1. **Program to Write Data to Memory (RAM)**
   - **Aim: Store the data 55H at memory location 2050H.**
   - **Assumptions: Memory location 2050H is within the interfaced RAM.**
   - **Assembly Code:**
   - Code snippet

```
  ORG 0000H       ; Program starts at 0000H (typical ROM location)

   LXI H, 2050H    ; Load 16-bit address 2050H into HL
   MVI A, 55H      ; Move immediate data 55H into Accumulator A
   MOV M, A        ; Move content of Accumulator (A) to memory location (HL)

   HLT            ; Halt processor
```

   -
   -
   - **Machine Code (Hexadecimal):**
     - **21 (LXI H opcode)**
     - **50 (Lower byte of address)**
     - **20 (Higher byte of address)**
     - **3E (MVI A opcode)**
     - **55 (Immediate data)**
     - **77 (MOV M, A opcode)**
     - **76 (HLT opcode)**
2. **Program to Read Data from Memory (RAM/ROM)**
   - **Aim: Read data from memory location 2050H and store it in register B.**
   - **Assumptions: Memory location 2050H has some data (e.g., from the previous write operation).**
   - **Assembly Code:**
   - Code snippet
```

```
ORG 0000H

 LXI H, 2050H    ; Load 16-bit address 2050H into HL
 MOV A, M        ; Move content of memory location (HL) to Accumulator (A)
 MOV B, A        ; Move content of Accumulator (A) to Register B

 HLT
```

-
-
- **Machine Code (Hexadecimal):**
  - **21**
  - **50**
  - **20**
  - **7E (MOV A, M opcode)**
  - **47 (MOV B, A opcode)**
  - **76**

3. **Program for Block Transfer of Data**
   - **Aim: Transfer 5 bytes of data from source addresses 2000H−2004H to destination addresses 2100H−2104H.**
   - **Assumptions: Both source and destination ranges are within interfaced RAM.**
   - **Assembly Code:**
   - Code snippet

```
ORG 0000H

 LXI H, 2000H    ; Load source starting address into HL
 LXI D, 2100H    ; Load destination starting address into DE
 MVI C, 05H      ; Initialize byte count to 5 in register C

LOOP:
 MOV A, M        ; Move data from source (HL) to Accumulator
 STAX D          ; Store Accumulator data to destination (DE)
 INX H           ; Increment HL to next source address
 INX D           ; Increment DE to next destination address
 DCR C           ; Decrement byte count
 JNZ LOOP        ; Jump to LOOP if C is not zero

 HLT
```

-
-
- **Machine Code (Hexadecimal):**
  - **21 (LXI H opcode)**
  - **00 20 (Source Address)**
  - **11 (LXI D opcode)**
  - **00 21 (Destination Address)**
  - **0E (MVI C opcode)**

        - **05 (Byte count)**
        - **7E (MOV A, M opcode)**
        - **12 (STAX D opcode)**
        - **23 (INX H opcode)**
        - **13 (INX D opcode)**
        - **0D (DCR C opcode)**
        - **C2 (JNZ opcode)**
        - **09 00 (Address of LOOP label - assuming LOOP is at 0009H)**
        - **76 (HLT opcode)**

Part C: Verification on 8085 Trainer Kit

1. **Power On the 8085 trainer kit.**
2. **Enter Machine Code: Using the kit's monitor program, navigate to the memory entry mode. Starting from address 0000H, enter the hexadecimal machine code for each program.**
    - **Pre-fill Source Data (for Block Transfer): Before running the block transfer program, manually enter some test data into memory locations 2000H to 2004H (e.g., 01H,02H,03H,04H,05H).**
3. **Verify Program Entry: Use the "Examine Memory" function to confirm that all opcodes and operands have been entered correctly.**
4. **Execute Programs:**
    - **For "Write Data to Memory":**
        - **Execute the program by entering its starting address (0000H) and pressing the "GO" key.**
        - **After execution, use "Examine Memory" to check the content of location 2050H.**
    - **For "Read Data from Memory":**
        - **Execute the program from 0000H.**
        - **After execution, use "Examine Registers" to check the content of the B register.**
    - **For "Block Transfer":**
        - **Execute the program from 0000H.**
        - **After execution, use "Examine Memory" to check the content of locations 2100H to 2104H.**
5. **Single-Stepping (Optional but Recommended for Understanding):**
    - **Load any of the programs.**
    - **Use the "Single Step" function (often labeled "STEP" or "S").**
    - **After each step, observe the changes in the Program Counter (PC), other registers (A, B, C, D, E, H, L), and memory contents. Pay attention to how the flags change, especially the Zero flag for JNZ instruction in the block transfer program.**

6. Observations

Record your observations during the execution of each program.

- **Observation for Program 1 (Write Data):**
    - **Before execution: Content of memory location 2050H was [Initial value, e.g., 00H or random].**
    - **After execution: Content of memory location 2050H is observed to be `55H`.**
    - **Conclusion: The `MOV M,A` instruction successfully wrote data to the specified RAM location.**
- **Observation for Program 2 (Read Data):**
    - **Before execution: Content of register B was [Initial value, e.g., 00H].**
    - **After execution: Content of register B is observed to be `55H`.**
    - **Conclusion: The `MOV A,M` instruction successfully read data from memory location 2050H into the Accumulator, and then it was transferred to register B.**
- **Observation for Program 3 (Block Transfer):**
    - **Before execution (Source Data in 2000H−2004H):**
        - **2000H: [e.g., 01H]**
        - **2001H: [e.g., 02H]**
        - **2002H: [e.g., 03H]**
        - **2003H: [e.g., 04H]**
        - **2004H: [e.g., 05H]**
    - **After execution (Destination Data in 2100H−2104H):**
        - **2100H: [Observed value, e.g., 01H]**
        - **2101H: [Observed value, e.g., 02H]**
        - **2102H: [Observed value, e.g., 03H]**
        - **2103H: [Observed value, e.g., 04H]**
        - **2104H: [Observed value, e.g., 05H]**
    - **Conclusion: The block transfer program successfully copied data from the source memory block to the destination memory block. The loop iterated correctly based on the byte count.**

7. Deliverables

1. **Designed Memory Map: (As presented in Section 5, Table 1).**
2. **Interfacing Schematic: (Detailed diagram showing connections between 8085, memory chips, and decoding logic, as conceptually described in Section 5 and to be drawn clearly).**
3. **Assembly Code for each program: (As presented in Section 5).**
4. **Machine Code for each program: (As presented in Section 5).**
5. **Observations and Results: (As recorded in Section 6).**

8. Conclusion

This experiment successfully demonstrated the principles of memory interfacing with the 8085 microprocessor. We learned to design a memory map, understand the necessity of address decoding, and implement interfacing logic. Furthermore, we gained practical experience in writing and executing assembly language programs to perform fundamental memory operations such as writing data to RAM, reading data

from memory (RAM/ROM), and transferring blocks of data, verifying these operations using the 8085 trainer kit's features. This understanding is foundational for designing and troubleshooting microprocessor-based systems.
