EXPERIMENT NO. 1

TITLE: Introduction to 8085 Microprocessor - Architecture and Basic Operations

OBJECTIVE:

To familiarize students with the 8085 microprocessor architecture, pin diagram, and the execution of basic data transfer and arithmetic instructions using an 8085 trainer kit or simulator.

SYLLABUS COVERAGE:

Overview of microcomputer systems and their building blocks, instruction sets of microprocessors (with examples of 8085).

---

1. THEORY / BACKGROUND

1.1 Overview of Microcomputer Systems and Their Building Blocks

A microcomputer system is a compact, cost-effective digital computing device that uses a microprocessor as its Central Processing Unit (CPU). It is designed for performing specific computational tasks and interacting with the external environment through input and output devices.

The fundamental components that form a microcomputer system are:

- **Microprocessor (CPU): This is the core computational and control unit. It is responsible for fetching instructions, decoding them, and executing operations, including arithmetic, logical comparisons, and controlling the flow of data between memory and I/O devices. Examples include the 8085, 8086, and ARM processors.**
- **Memory Unit: This unit serves as the storage area for both program instructions and data that the CPU needs to access.**
    - **Read-Only Memory (ROM): Stores permanent programs, such as the initial boot-up instructions (monitor program) for the system. Data in ROM is non-volatile, meaning it retains its contents even when power is off.**
    - **Random Access Memory (RAM): Used for temporary storage of programs and data that are actively being processed by the CPU. RAM is volatile, so its contents are lost when the power supply is disconnected.**
- **Input/Output (I/O) Unit: These devices enable the microcomputer to communicate with the outside world.**
    - **Input Devices: Devices such as keyboards, switches, or sensors, that allow external data or commands to be fed into the computer.**

    - **Output Devices: Devices such as LED displays, printers, or motor drivers, that present the results of computations or control external hardware based on the computer's operations.**

System Buses: Communication pathways connecting the CPU to memory and I/O devices. These are groups of parallel wires that carry specific types of information.

- **Address Bus: A unidirectional bus through which the CPU specifies the unique memory location or I/O port address it wishes to access. The number of lines on the address bus determines the maximum memory capacity the CPU can address. For instance, the 8085 has a 16-bit address bus, allowing it to access 2^{16}=65,536 unique memory locations (64 KB).**
- **Data Bus: A bidirectional bus used for transferring data between the CPU, memory, and I/O devices. The width of the data bus (number of lines) indicates how many bits of data the CPU can process at one time. The 8085 has an 8-bit data bus, meaning it handles 8 bits (one byte) of data per transfer.**
- **Control Bus: A collection of various individual control signals that synchronize and manage the operations of all components within the microcomputer system. These signals dictate the nature of the current operation (e.g., memory read, I/O write), timing, and direction of data flow. Examples include the Read (RD), Write (WR), and Memory/IO (IO/M) signals.**

The CPU constantly performs the fetch-decode-execute cycle. It retrieves an instruction from memory using the address bus, reads the instruction via the data bus, interprets its meaning, and then performs the required operation. This operation might involve reading or writing data to memory or an I/O device, all orchestrated by the control bus.

1.2 Introduction to 8085 Microprocessor

The Intel 8085 is an 8-bit microprocessor, signifying its capability to process 8 bits of data in parallel. It was a foundational component in the development of microcomputer technology and remains widely used for educational purposes due to its straightforward architecture. It operates on a single +5V power supply and is housed in a 40-pin Dual-In-line Package (DIP).

Key Features of 8085:

- **8-bit Processor: Operates on 8-bit data units.**
- **16-bit Address Bus: Enables addressing up to 64 KB of memory.**
- **40-pin DIP Package: Standard physical form factor.**
- **Single +5V Power Supply: Simplified power requirements.**
- **On-chip Clock Generator: Requires an external crystal or RC network for timing.**
- **Internal Interrupt Controller: Manages various interrupt requests.**

1.2.1 8085 Architecture:

The internal structure of the 8085 microprocessor comprises several specialized functional units:

- **Arithmetic Logic Unit (ALU): This unit is responsible for executing all arithmetic operations (addition, subtraction, increment, decrement) and logical operations (AND, OR, XOR, NOT, comparisons). The results of these operations are typically stored in the Accumulator register.**
- **Registers: These are small, fast memory locations within the CPU used for temporary storage of data and addresses during program execution.**
    - **Accumulator (A): An 8-bit register that is central to almost all arithmetic and logical operations. One operand for an operation is usually in the Accumulator, and the result is stored back in it.**
    - **General Purpose Registers (B, C, D, E, H, L): These are six 8-bit registers that can be used for general-purpose data storage. They can also be paired to handle 16-bit data: BC, DE, and HL. The HL pair is particularly significant as it often serves as a 16-bit memory pointer, holding the address of a memory location.**
    - **Flag Register: An 8-bit register where only five bits are actively used. Each of these five bits is a "flag" that reflects the status of the most recent arithmetic or logical operation. A flag is set (1) or reset (0) based on the result.**
        - **S (Sign Flag): Set to 1 if the most significant bit (MSB, D7) of the result is 1 (indicating a negative number in signed arithmetic); otherwise, 0.**
        - **Z (Zero Flag): Set to 1 if the result of the operation is 0; otherwise, 0.**
        - **AC (Auxiliary Carry Flag): Set to 1 if there is a carry from bit 3 to bit 4 during an arithmetic operation (primarily used for BCD - Binary Coded Decimal arithmetic); otherwise, 0.**
        - **P (Parity Flag): Set to 1 if the result contains an even number of set bits (1s); otherwise, 0 (for odd parity).**
        - **C (Carry Flag): Set to 1 if there is a carry out from the most significant bit (D7) in addition, or if a borrow is required in subtraction; otherwise, 0.**
    - **Program Counter (PC): A 16-bit register that holds the memory address of the *next* instruction byte to be fetched by the CPU. After each instruction byte is fetched, the PC is automatically incremented.**
    - **Stack Pointer (SP): A 16-bit register that contains the 16-bit memory address of the top of the stack. The stack is a dedicated area in RAM used for temporary storage of addresses during subroutine calls (to return to the main program) or for saving register contents during interrupts.**
- **Instruction Decoder and Machine Cycle Encoding: This unit interprets the fetched instruction's opcode and generates the necessary internal control signals for its execution.**
- **Timing and Control Unit: This unit generates the timing and control signals required for all internal operations of the microprocessor and for controlling external devices.**

- **Interrupt Control: Manages various hardware interrupt requests (TRAP, RST7.5, RST6.5, RST5.5, INTR) to allow the CPU to respond to external events.**
- **Serial I/O Control: Provides dedicated pins (SID - Serial Input Data, SOD - Serial Output Data) for serial data communication.**

1.2.2 Key 8085 Pin Descriptions:

The 8085 is packaged in a 40-pin integrated circuit. Understanding the function of its key pins is essential for interfacing.

- **AD0-AD7 (Pins 12-19): These are the lower 8 bits of the address bus (A0-A7) during the initial part of a machine cycle, and then function as the 8-bit bidirectional data bus (D0-D7) during the rest of the cycle. This time-sharing is called multiplexing.**
- **A8-A15 (Pins 21-28): These pins form the higher 8 bits of the address bus. They are not multiplexed and remain as address lines throughout the machine cycle.**
- **ALE (Address Latch Enable, Pin 30): This is a positive-going pulse emitted by the 8085 at the start of each machine cycle. Its falling edge signals that the address (A0-A7) on AD0-AD7 is valid and should be latched by external circuitry to separate it from data.**
- **RD (Read, Pin 32): An active-low output signal. When low, it indicates that the CPU is performing a memory read or I/O read operation.**
- **WR (Write, Pin 31): An active-low output signal. When low, it indicates that the CPU is performing a memory write or I/O write operation.**
- **IO/M (I/O / Memory, Pin 34): A status output signal. When high (1), it indicates an I/O operation; when low (0), it indicates a memory operation. This signal, combined with RD and WR, specifies the type of access.**
- **S0, S1 (Status Signals, Pins 33, 29): These two output pins provide information about the type of machine cycle currently being executed by the 8085 (e.g., Opcode Fetch, Memory Read, I/O Write).**
- **RESET IN (Pin 36): An active-low input. When asserted (brought low), it clears the Program Counter to 0000H and resets the 8085, bringing it to a known initial state.**
- **RESET OUT (Pin 3): An active-high output signal that goes high when the 8085 is being reset (e.g., via RESET IN). This signal is used to reset other peripheral devices connected to the system.**
- **CLK OUT (Pin 37): An output clock signal that provides a frequency equal to half the crystal frequency connected to X1 and X2. This can be used to synchronize other components in the system.**
- **X1, X2 (Pins 1, 2): These pins are connected to an external crystal or RC network. An internal oscillator generates the clock signal based on this external component.**
- **Vcc (Pin 40): The power supply pin, requiring a +5V DC supply.**
- **Vss (Pin 20): The ground reference pin.**
- **INTR (Interrupt Request, Pin 10): A general-purpose, maskable interrupt input. The CPU checks this pin during the last clock cycle of each instruction.**

- **INTA (Interrupt Acknowledge, Pin 11): An active-low output signal. When low, it indicates that the 8085 has acknowledged an INTR request and is ready to receive the interrupt vector.**
- **RST 7.5, RST 6.5, RST 5.5 (Restart Interrupts, Pins 7, 8, 9): These are maskable, vectored interrupt inputs. They have higher priority than INTR and cause the program to jump to specific predefined memory locations.**
- **TRAP (Pin 6): A non-maskable, highest priority interrupt input. It cannot be disabled by software and is typically used for critical events like power failure.**
- **HOLD (Pin 39): An input signal to request the 8085 to release its address, data, and control buses. This is typically used by DMA (Direct Memory Access) controllers to take control of the buses for high-speed data transfer.**
- **HLDA (Hold Acknowledge, Pin 38): An output signal. When high, it indicates that the 8085 has relinquished control of its buses in response to a HOLD request.**

1.3 8085 Instruction Set (Basic Data Transfer and Arithmetic Operations)

The 8085 instruction set is the set of all commands that the microprocessor understands and can execute. Each instruction consists of an Opcode (the operation code, specifying what to do) and an Operand (the data or address on which the operation is to be performed).

Instruction Format: `OPCODE OPERAND`

Addressing Modes: These define how the CPU locates the data (operand) required for an instruction.

- **Immediate Addressing: The data itself is provided directly within the instruction.**
  - **Example: `MVI A, 30H` (The value 30H is the operand).**
- **Register Addressing: The operand is located in one of the 8085's internal registers.**
  - **Example: `MOV B, A` (Register A's content is the operand).**
- **Direct Addressing: The instruction includes the exact 16-bit memory address where the operand is located.**
  - **Example: `LDA 2000H` (The address 2000H is explicitly given).**
- **Register Indirect Addressing: The 16-bit address of the operand is held in a register pair (typically HL). The instruction refers to this register pair, and the CPU then accesses the memory location pointed to by that pair.**
  - **Example: `MOV M, A` (M refers to the memory location whose address is in the HL register pair).**

1.3.1 Data Transfer Instructions:

These instructions are used to copy data from a source to a destination. The content of the source location remains unchanged.

- **MOV (Move): Copies data between two registers or between a register and a memory location.**
    - `MOV Rd, Rs`**: Copies content of source register** `Rs` **to destination register** `Rd`**.**
        - **Example:** `MOV B, A` **(Copies A's content to B).**
    - `MOV Rd, M`**: Copies content of memory (pointed by HL) to destination register** `Rd`**.**
        - **Example:** `MOV C, M` **(Copies data from memory location** `[HL]` **to C).**
    - `MOV M, Rs`**: Copies content of source register** `Rs` **to memory (pointed by HL).**
        - **Example:** `MOV M, B` **(Copies B's content to memory location** `[HL]`**).**
- **MVI (Move Immediate): Loads immediate 8-bit data into a specified register or memory location.**
    - `MVI R, Data`**: Loads** `Data` **into register** `R`**.**
        - **Example:** `MVI A, 45H` **(A becomes 45H).**
    - `MVI M, Data`**: Loads** `Data` **into memory location pointed by HL.**
        - **Example:** `MVI M, 60H` **(Memory at** `[HL]` **becomes 60H).**
- **LXI (Load Register Pair Immediate): Loads immediate 16-bit data into a specified register pair (BC, DE, or HL).**
    - `LXI Rp, 16-bit Data`**: Loads** `16-bit Data` **into register pair** `Rp`**.**
        - **Example:** `LXI H, 2050H` **(H becomes 20H, L becomes 50H).**
- **LDA (Load Accumulator Direct): Loads the Accumulator with the 8-bit data from a specified 16-bit memory address.**
    - `LDA 16-bit Address`**:**
        - **Example:** `LDA 3000H` **(A becomes content of memory location 3000H).**
- **STA (Store Accumulator Direct): Stores the 8-bit content of the Accumulator into a specified 16-bit memory address.**
    - `STA 16-bit Address`**:**
        - **Example:** `STA 4000H` **(Memory location 4000H gets content of A).**

1.3.2 Arithmetic Instructions:

These instructions perform basic arithmetic operations (addition, subtraction, increment, decrement) and significantly affect the status flags.

- **ADD (Add Register to Accumulator): Adds the content of a specified register to the Accumulator. The result is stored in the Accumulator.**
    - `ADD R`**: A = A + R.**
        - **Example:** `ADD B` **(A = A + B).**
        - **Numerical Example:**
            - **Initial: A = 10H, B = 05H**
            - **Instruction:** `ADD B`

        - **Result: A = 15H. Flags: S=0, Z=0, AC=0, P=1, C=0.**
- **ADI (Add Immediate to Accumulator): Adds immediate 8-bit data to the Accumulator. The result is stored in the Accumulator.**
    - **`ADI Data`: A = A + Data.**
        - **Example: `ADI 0FH` (A = A + 0FH).**
        - **Numerical Example (with Carry):**
            - **Initial: A = F0H, B = 20H**
            - **Instruction: `ADI 20H` (Assuming we use ADI directly here for simplicity, though the example implies ADD B from prev.)**
            - **Calculation: F0H (240) + 20H (32) = 110H (272).**
            - **Result: A = 10H (lower 8 bits). Flags: S=0, Z=0, AC=0, P=1, C=1. (Carry flag set due to overflow).**
- **SUB (Subtract Register from Accumulator): Subtracts the content of a specified register from the Accumulator. The result is stored in the Accumulator.**
    - **`SUB R`: A = A - R.**
        - **Example: `SUB C` (A = A - C).**
        - **Numerical Example (with Borrow):**
            - **Initial: A = 05H, C = 10H**
            - **Instruction: `SUB C`**
            - **Calculation: 05H - 10H. In 8-bit 2's complement, this becomes 05H + (2's complement of 10H) = 05H + F0H = F5H.**
            - **Result: A = F5H. Flags: S=1, Z=0, AC=0, P=0, C=1. (Carry flag set, indicating a borrow).**
- **SUI (Subtract Immediate from Accumulator): Subtracts immediate 8-bit data from the Accumulator. The result is stored in the Accumulator.**
    - **`SUI Data`: A = A - Data.**
        - **Example: `SUI 08H` (A = A - 08H).**
- **INR (Increment Register/Memory): Increments the content of a specified register or memory location by 1.**
    - **`INR R`: R = R + 1.**
        - **Example: `INR B` (If B was 0FH, it becomes 10H).**
    - **`INR M`: M = M + 1 (Increments content of memory location pointed by HL).**
        - **Example: `INR M` (If HL = 2000H and memory[2000H] = 05H, it becomes 06H).**
- **DCR (Decrement Register/Memory): Decrements the content of a specified register or memory location by 1.**
    - **`DCR R`: R = R - 1.**
        - **Example: `DCR C` (If C was 10H, it becomes 0FH).**
    - **`DCR M`: M = M - 1 (Decrements content of memory location pointed by HL).**
        - **Example: `DCR M` (If HL = 2000H and memory[2000H] = 05H, it becomes 04H).**

2. APPARATUS / SOFTWARE REQUIRED

- **8085 Microprocessor Trainer Kit (e.g., VMC-8500, Dyalog-85)**
  OR
  8085 Microprocessor Simulator Software (e.g., GNUSIM8085, Online 8085 Simulator)
- **DC Power Supply (typically +5V, for hardware kit)**
- **Connecting wires (if any external components are to be connected, though not needed for this basic experiment on trainer kit)**
- **8085 Trainer Kit User Manual (for specific commands)**
- **Personal Computer (for simulator use)**

---

3. PROCEDURE

Follow these steps to perform the experiments using an 8085 Trainer Kit (or adapt for a simulator by referring to its specific commands):

3.1 System Setup and Power On

1. **Place the 8085 Microprocessor Trainer Kit on a stable surface.**
2. **Connect the DC power supply to the kit and switch on the power. Observe the initial display (often showing "8085>" or "MONITOR" indicating readiness).**
3. **Press the "RESET" button on the trainer kit. This initializes the microprocessor and loads the monitor program, setting the Program Counter to a known starting address (usually 0000H or the monitor's entry point).**

3.2 Program Entry

1. **Select Starting Address: Use the appropriate command on your trainer kit (e.g., `EXAM MEM` or `GO MEM`) followed by the 16-bit hexadecimal address where you wish to begin entering your program (e.g., `2000`).**
2. **Enter Opcode and Operand: For each instruction in your assembly program, enter its corresponding 2-digit hexadecimal opcode using the keypad. If the instruction has an operand (data or address), enter it immediately after the opcode. The trainer kit's monitor program will automatically display the next memory address after each byte is entered.**
   - **Example: To enter `MVI A, 15H` (Opcode: `3E`, Operand: `15`):**
     - **At address `2000`, enter `3E`. The display will advance to `2001`.**
     - **At address `2001`, enter `15`. The display will advance to `2002`.**
   - **Example: To enter `ADD B` (Opcode: `80`):**
     - **At address `2004`, enter `80`. The display will advance to `2005`.**
3. **End Program Entry: After entering all instructions for your program, press the designated key to exit memory entry mode (e.g., `RESET`, `NEXT`, or `ENTER`, as per your kit's manual).**

3.3 Program Execution

1. **Execute Command: Use the `GO` or `EXEC` command on your trainer kit followed by the 16-bit starting address of your program.**
    - **Example: To execute a program starting at 2000H, type `GO 2000` and press `ENTER`.**
2. **Single-Stepping (Optional): For debugging and observing changes instruction by instruction, some kits offer a `STEP` command. This executes one instruction at a time, allowing you to examine registers and memory after each step.**

3.4 Observation

1. **Examine Registers: After program execution (or after each step in single-stepping mode), use the `EXAM REG` or `REG` command (refer to your kit's manual) to view the current contents of the Accumulator, other general-purpose registers (B, C, D, E, H, L), the Program Counter (PC), Stack Pointer (SP), and the Flag Register. Note down these values.**
2. **Examine Memory: Use the `EXAM MEM` or `DISPLAY MEM` command followed by a 16-bit memory address (e.g., `EXAM MEM 2050`) to view the contents of specific memory locations where your program stored data. You can usually press a `NEXT` or `INC` key to view subsequent memory locations. Note down the contents.**

---

4. PROGRAMS TO BE EXECUTED

For each program, record the actual observed values in the "Observations" section after execution.

4.1 Program 1: Basic Data Transfer Operations

Objective: To move an immediate 8-bit data into the Accumulator, then copy it to Register B, and finally store the content of Register B into a specific memory location (2050H).

Assembly Code:

```
Code snippet

; Program to demonstrate data transfer
; Starting Address: 2000H

ORG 2000H       ; Assembler directive: Program starts at address 2000H

MVI A, 15H      ; Move Immediate 15H into Accumulator (A = 15H)
MOV B, A        ; Move content of Accumulator A to Register B (B = A)
LXI H, 2050H    ; Load HL pair with 16-bit address 2050H (HL = 2050H)
MOV M, B        ; Move content of Register B to memory location pointed by HL ([2050H] = B)
HLT             ; Halt processor execution
```

Memory Layout (Opcode and Operand in Hex):

| Address | Hex Code | Instruction | Comments |

| :------ | :------- | :----------------- | :-- |

| 2000H | 3E | MVI A, 15H | Opcode for MVI A |

| 2001H | 15 | | Operand (Data 15H) |

| 2002H | 47 | MOV B, A | Opcode for MOV B, A |

| 2003H | 21 | LXI H, 2050H | Opcode for LXI H |

| 2004H | 50 | | Lower byte of address (50H) |

| 2005H | 20 | | Higher byte of address (20H) |

| 2006H | 70 | MOV M, B | Opcode for MOV M, B |

| 2007H | 76 | HLT | Opcode for HLT |

Expected Register and Memory States (After Execution of `HLT`):

- **A Register: 15H**
- **B Register: 15H**
- **H Register: 20H**
- **L Register: 50H**
- **Memory Location 2050H: 15H**
- **Program Counter (PC): 2008H (Points to the next location after HLT)**
- **Flag Register: (State depends on prior operations, but for this specific sequence, flags are generally unaffected by MOV/LXI if no arithmetic is involved, might show default or previous state).**

4.2 Program 2: Basic Arithmetic Operation (Addition)

Objective: To add two 8-bit numbers (05H and 03H), store the result in the Accumulator, then save it to a memory location (2060H), and observe the flag status.

Assembly Code:

```
Code snippet

; Program to demonstrate addition and flag changes
; Starting Address: 2000H

ORG 2000H

MVI A, 05H      ; Load Accumulator A with immediate data 05H (A = 05H)
MVI B, 03H      ; Load Register B with immediate data 03H (B = 03H)
ADD B           ; Add content of Register B to Accumulator A (A = A + B)
```

```
STA 2060H       ; Store content of Accumulator A into memory location 2060H
HLT             ; Halt processor execution
```

Memory Layout:

Address	Hex Code	Instruction
2000H	3E	MVI A, 05H
2001H	05	
2002H	06	MVI B, 03H
2003H	03	
2004H	80	ADD B
2005H	32	STA 2060H
2006H	60	
2007H	20	
2008H	76	HLT

Expected Register and Memory States (After Execution of `HLT`):

- **A Register: 08H (05H + 03H)**
- **B Register: 03H**
- **Memory Location 2060H: 08H**
- **Program Counter (PC): 2009H**
- **Flag Register Status (after `ADD B` where A=05H, B=03H, Result A=08H):**
    - **S (Sign Flag): 0 (Result 08H is positive)**
    - **Z (Zero Flag): 0 (Result 08H is not zero)**
    - **AC (Auxiliary Carry Flag): 0 (No carry from bit 3 to bit 4)**
    - **P (Parity Flag): 0 (Result 08H = 00001000b, has one '1' bit, which is odd parity)**
    - **C (Carry Flag): 0 (No carry out from bit 7)**

4.3 Program 3: Basic Arithmetic Operation (Subtraction with Borrow)

Objective: To subtract a larger number from a smaller number (05H - 10H), store the result in memory (2070H), and observe the specific flag status, particularly the Carry/Borrow flag.

Assembly Code:

Code snippet

```
; Program to demonstrate subtraction and flag changes (with borrow)
; Starting Address: 2000H

ORG 2000H

MVI A, 05H      ; Load Accumulator A with immediate data 05H (A = 05H)
MVI B, 10H      ; Load Register B with immediate data 10H (B = 10H)
SUB B           ; Subtract content of Register B from Accumulator A (A = A - B)
STA 2070H       ; Store content of Accumulator A into memory location 2070H
HLT             ; Halt processor execution
```

Memory Layout:

Address	Hex Code	Instruction
2000H	3E	MVI A, 05H
2001H	05	
2002H	06	MVI B, 10H
2003H	10	
2004H	90	SUB B
2005H	32	STA 2070H
2006H	70	
2007H	20	
2008H	76	HLT

Expected Register and Memory States (After Execution of `HLT`):

- **A Register: F5H (05H - 10H = F5H in 8-bit 2's complement, representing -11 decimal)**
- **B Register: 10H**
- **Memory Location 2070H: F5H**
- **Program Counter (PC): 2009H**
- **Flag Register Status (after `SUB B` where A=05H, B=10H, Result A=F5H):**
    - **S (Sign Flag): 1 (Result F5H is negative, MSB is 1)**
    - **Z (Zero Flag): 0 (Result F5H is not zero)**
    - **AC (Auxiliary Carry Flag): 0 (No intermediate borrow)**
    - **P (Parity Flag): 0 (Result F5H = 11110101b, has five '1' bits, which is odd parity)**

- **C (Carry Flag): 1 (Carry flag is set, indicating a borrow occurred from the most significant bit, which happens when a larger number is subtracted from a smaller one).**

---

5. OBSERVATIONS AND RESULTS

Record the actual observed values from the 8085 Trainer Kit/Simulator after executing each program. Compare them with the expected results.

5.1 Program 1 Observations:

Register/Memory Location	Expected Value	Observed Value
A Register	15H	
B Register	15H	
H Register	20H	
L Register	50H	
Memory Location 2050H	15H	
Program Counter (PC)	2008H	

5.2 Program 2 Observations:

Register/Memory Location	Expected Value	Observed Value
A Register	08H	

B Register	**03H**	
Memory Location 2060H	**08H**	
Program Counter (PC)	**2009H**	
Flag Register	**Expected**	**Observed**
Sign (S)	**0**	
Zero (Z)	**0**	
Aux. Carry (AC)	**0**	
Parity (P)	**0**	
Carry (C)	**0**	

5.3 Program 3 Observations:

Register/Memory Location	Expected Value	Observed Value
A Register	**F5H**	
B Register	**10H**	
Memory Location 2070H	**F5H**	

Program Counter (PC)	2009H	
Flag Register	**Expected**	**Observed**
Sign (S)	1	
Zero (Z)	0	
Aux. Carry (AC)	0	
Parity (P)	0	
Carry (C)	1	

---

6. CONCLUSION

Conclude what you have learned from this experiment. Summarize your observations regarding the 8085 architecture, the functionality of basic data transfer and arithmetic instructions, and how different operations affect the CPU's internal registers and status flags. Comment on any discrepancies between expected and observed results, and provide possible reasons.

---

7. VIVA VOCE QUESTIONS

1. **What are the three main buses in an 8085 based microcomputer system, and what is the function of each?**
2. **Explain the concept of address/data multiplexing in the 8085. Which signal is used to de-multiplex this bus?**
3. **Name the general-purpose registers in the 8085. How can they be used as register pairs?**
4. **Describe the function of the Accumulator and the Program Counter in the 8085.**
5. **List all the flags in the 8085 Flag Register. What does it mean if the Zero flag is set after an operation? What if the Carry flag is set?**
6. **Differentiate between `MOV A, B` and `MVI A, 20H` instructions.**

7. **If the Accumulator contains 80H and register B contains 10H, what will be the content of the Accumulator and the status of the Sign and Carry flags after `ADD B` instruction?**
8. **Why is the `HLT` instruction necessary at the end of a program?**
9. **What is the purpose of the `LXI` instruction? Give an example.**
10. **What is the difference between an opcode and an operand?**
