GOVERNMENT POLTECHNIC DARBHANGA

Department of Electronics Engineering

4th Semester:- Microcontroller and its Applications

Lab Manual Subject Code [P2421402]



LIST OF EXPERIMENTS

S.NO.	Experiment Name
1	Verification of the features of 8085 Trainer Kit
2	ALP based on Data transfer Instructions & Arithmetic Instructions(addition) in 8085
3	ALP based on Data transfer Instructions and arithmetic Instructions(subtraction) in 8085
4	Verification of the features of 8051 Trainer Kit
5	ALP based on Data transfer Instructions & Arithmetic Instructions(addition) in 8051
6	ALP based on Data transfer Instructions and arithmetic Instructions(subtraction) in 8051
7	Write an ALP to generate delay using Timer
8	Interfacing 7 segment display with 8051

List of Major Laboratory Equipment, Tools and Software:

S. NO.	Name of Equipment, Tools, and Software	Broad Specifications	Relevant Experiment/Practical Number
1	Microcontroller Trainer kit	Single board systems with 8K RAM, ROM memory with battery backup, 16X4,16 X2, LCD display, PC keyboard interfacing facility, Hex keypad facility, single user cross c-compiler, RS-232, USB, interfacing facility with built in power supply	All
2	PC with microcontroller simulation software	Desktop PC (above I5& 16GB RAM) with microcontroller simulation software	All
3	CRO/DSO	Bandwidth AC l0Hz ~ 20MHz (-3dB). DC~ 20MHz (-3dB), Xl0 Probe	All
4	Stepper Motor	50/100 RPM	10
5	Trainer board	Keyboard 4*4 trainer board	7 -11
6	7- segment LED Display	7-segment LED Display: - 0.56 in 1-digit, common anode/common cathode	8
7	Multimeter	Digital	8 -11
8	Trainer board	ADC (0808) trainer board	11
9	Trainer board	DAC (0808) trainer board	11
10	Trainer board	LCD trainer board	8
11	Sensors	LM35, ADXL345, AD590, BMP085, DT11, NTC, IIS3DWBTR, etc.	7

Experiment:01

Verification of the features of 8085 Trainer Kit

Aim

To study and verify the various features and functions of the 8085 microprocessor trainer kit.

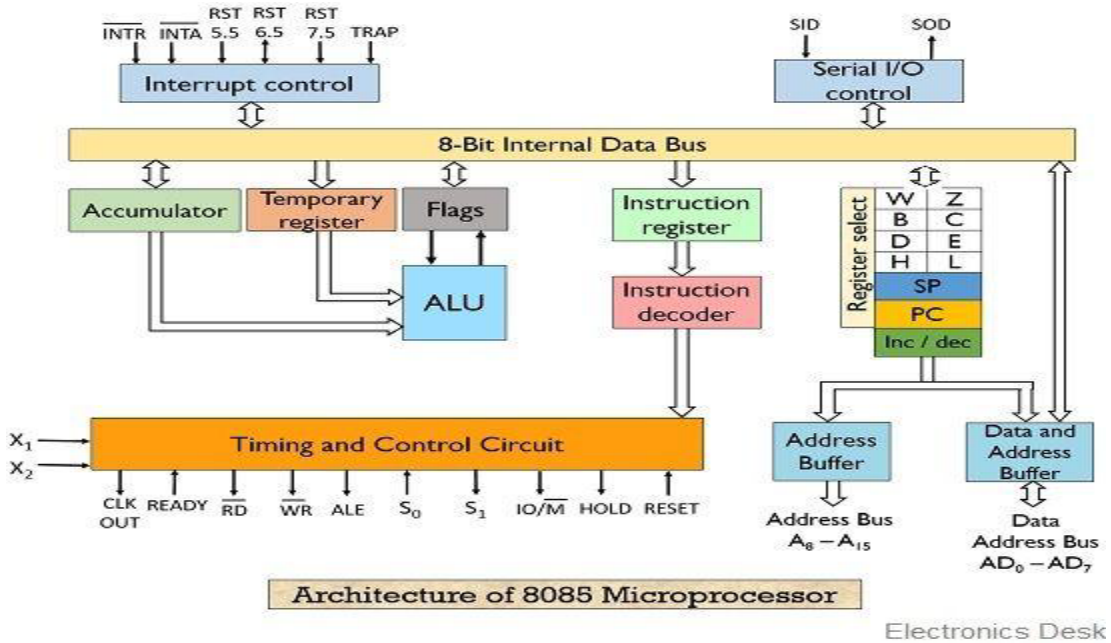
Apparatus Required

- 8085 Microprocessor Trainer Kit
- Power supply
- Connecting wires (if required)

Theory

The 8085 Trainer Kit is a compact educational system designed to learn microprocessor architecture, programming, and interfacing. It consists of CPU, memory, input/output devices, and debugging facilities that allow users to write, execute, and test programs.

Features to be Verified



1. Microprocessor (CPU)

- Based on Intel 8085 (8-bit processor)
- Performs arithmetic and logical operations
  ✔ Verified by executing simple programs

2. Memry (RAM & ROM)

- RAM: Used to store user programs (temporary)
- ROM: Stores monitor program
  ✔ Verified by storing and retrieving data from memory locations

3. Monitor Program

- Allows program entry, execution, debugging
  ✔ Verified by:
- Entering program using keypad
- Executing step-by-step

4. Keyboard / Keypad

- Used to enter data and instructions
  ✔ Verified by entering hexadecimal codes

5. Display Unit

- 7-segment LED / LCD display
  ✔ Verified by observing:
- Address
- Data
- Execution result

6. Input / Output Ports

- Parallel I/O (using 8255)
- Serial I/O (SID/SOD lines)
  ✔ Verified by:
- Interfacing LEDs / switches
- Observing output changes

7. Timing and Control Unit

- Provides clock signal
  ✔ Verified by:
- Program execution timing

- Single-step operation

8. Interrupt System

- Handles external/internal interrupts
  ✔ Verified using interrupt-based programs

9. Debugging Features

- Single step execution
- Breakpoints
- Memory examine
  ✔ Verified by running programs step-by-step and checking registers

10. Interfacing Capability

- Supports ADC, DAC, stepper motor, etc.
  ✔ Verified by connecting external devices (if available)

11. Serial Communication

- Interface with PC via RS-232
  ✔ Verified by uploading/downloading programs

12. Power Supply

- Typically +5V regulated supply
  ✔ Verified by powering the kit

Procedure

1. Switch ON the trainer kit
2. Observe display initialization
3. Enter a simple program (e.g., addition)
4. Execute program using:
   - Run mode
   - Single-step mode
5. Verify output in registers/display
6. Test memory by storing and reading values
7. Check I/O ports using LEDs or switches

Result

All the features of the 8085 Trainer Kit were successfully verified, including CPU operation, memory access, I/O interfacing, and debugging functions.

Precautions

- Ensure proper power supply
- Avoid wrong connections
- Enter correct hexadecimal codes
- Reset kit before new program

Experiment:02

ALP based on Data transfer Instructions & Arithmetic Instructions (addition) in 8085

Aim

To write and execute an **Assembly Language Program (ALP)** using data transfer and arithmetic (addition) instructions in 8085 microprocessor.

Theory

- **Data Transfer Instructions**: Used to move data between registers, memory, and accumulator
  (e.g., MOV, MVI, LDA, STA)
- **Arithmetic Instructions**: Used to perform operations like addition
  (e.g., ADD, ADI)

Program 1: Addition of Two 8-bit Numbers (Registers)

Algorithm

1. Load first number into accumulator
2. Load second number into register
3. Add the numbers
4. Store the result

ALP Code

```
MVI A, 05H      ; Load 1st number into Accumulator
MVI B, 03H      ; Load 2nd number into Register B
ADD B           ; A = A + B
STA 2000H       ; Store result at memory location 2000H
HLT             ; Stop
```

📊 Explanation

- MVI → Data transfer instruction
- ADD → Arithmetic instruction
- Result = **08H** stored at 2000H

Program 2: Addition of Two Numbers from Memory

🔷 Algorithm

1. Load first number from memory
2. Move second number to register
3. Add
4. Store result

💻 ALP Code

```
LDA 2000H      ; Load first number into A
MOV B, A       ; Save in B
LDA 2001H      ; Load second number into A
ADD B          ; Add both numbers
STA 2002H      ; Store result
HLT
```

📊 Explanation

- LDA, STA, MOV → Data transfer
- ADD → Arithmetic
- Result stored at **2002H**

Program 3: Addition of Two 8-bit Numbers with Carry

💻 ALP Code

```
MVI A, FFH     ; Load first number
MVI B, 02H     ; Load second number
ADD B          ; Add
STA 2000H      ; Store result
MVI A, 00H     ; Clear A
ADC A          ; Add carry
STA 2001H      ; Store carry
HLT
```

Result

The ALP was successfully executed and verified using data transfer and arithmetic instructions for addition.

Precautions

- Use correct memory addresses
- Enter hex values carefully
- Reset before execution

Experiment:03

ALP based on Data transfer Instructions and arithmetic Instructions (subtraction) in 8085

Aim

To write and execute an **Assembly Language Program (ALP)** for subtraction using data transfer and arithmetic instructions in 8085.

Theory

- **Data Transfer Instructions**: `MOV`, `MVI`, `LDA`, `STA`
- **Arithmetic Instructions (Subtraction)**: SUB, SUI, SBB

☞ Subtraction in 8085 is performed using the **Accumulator (A)**.
☞ Result affects flags like **Carry (borrow), Zero, Sign**.

Program 1: Subtraction of Two 8-bit Numbers (Registers)

8085 ASSEMBLY LANGUAGE PROGRAMMING

2) PROGRAM TO ADD OR SUBTRACT TWO 8 BIT NUMBERS

Sl no.	ADDRESS	HEX CODE	LABEL	MNEMONICS	COMMENTS
1	8000	3E, XX		MVI A, XX	Move immediately to [A] the data XX
2	8002	06, YY		MVI B, YY	Move immediately to [B] the data YY
3	8004	80/(90)		ADD B/SUB B	Add(or subtract) [B] with [A]
4	8005	32,00,F0		STA F000	Store contents of [A] in specified location
5	8008	D2,10,80		JNC (Ahead 1)	Jump on no carry to specified location
6	800B	3E,01		MVIA, 01	Move immediately to [A] the data$(01)_H$
7	800D	C3,12,80		JMP(Ahead 2)	Jump unconditionally ahead
8	8010	3E,00	Ahead 1	MVIA, 00	Move immediately to [A] the data$(00)_H$
9	8012	32,01,F0	Ahead 2	STA F001	Store contents of [A] in specified location
10	8015	CF		RST1	Restart

INPUT : XX, YY
1) 2B, 1A
2) 2B, FA

OUTPUT

TRIAL	Addition (XX+YY)	Subtraction (XX-YY)
ADDRESS	F001: F000	F001: F000
1	00 : 45	00 : 25
2	01 : 25	01 : 31

K. ADISESHA, Presidency college 2

Algorithm

1. Load first number into accumulator
2. Load second number into register
3. Subtract register from accumulator
4. Store result

ALP Code

```
MVI A, 09H     ; Load first number
MVI B, 04H     ; Load second number
SUB B          ; A = A - B
STA 2000H      ; Store result at 2000H
HLT            ; Stop
```

Explanation

- MVI → Data transfer
- SUB → Arithmetic subtraction
- Result = **05H** stored at 2000H

Program 2: Subtraction of Two Numbers from Memory

Algorithm

1. Load first number from memory
2. Move it to register
3. Load second number
4. Subtract
5. Store result

ALP Code

```
LDA 2000H      ; Load first number
MOV B, A       ; Store in B
LDA 2001H      ; Load second number
SUB B          ; A = A - B
STA 2002H      ; Store result
HLT
```

📊 Explanation

- Data transfer: `LDA`, `MOV`, `STA`
- Arithmetic: `SUB`
- Result stored at **2002H**

Program 3: Subtraction with Borrow

💻 ALP Code

```
MVI A, 05H     ; Load first number
MVI B, 09H     ; Load second number
SUB B          ; A = A - B (borrow occurs)
STA 2000H      ; Store result
MVI A, 00H     ; Clear A
SBB A          ; Store borrow (carry flag)
STA 2001H      ; Store borrow
HLT
```

Result

The subtraction operation was successfully performed using data transfer and arithmetic instructions, and results were stored in memory.

Precautions

- Ensure correct order: **A – Register**
- Check carry flag for borrow
- Enter correct hex values
- Reset before execution

Experiment:04

Verification of the features of 8051 Trainer Kit

Aim

To study and verify the various features of the **8051 Microcontroller Trainer Kit**.

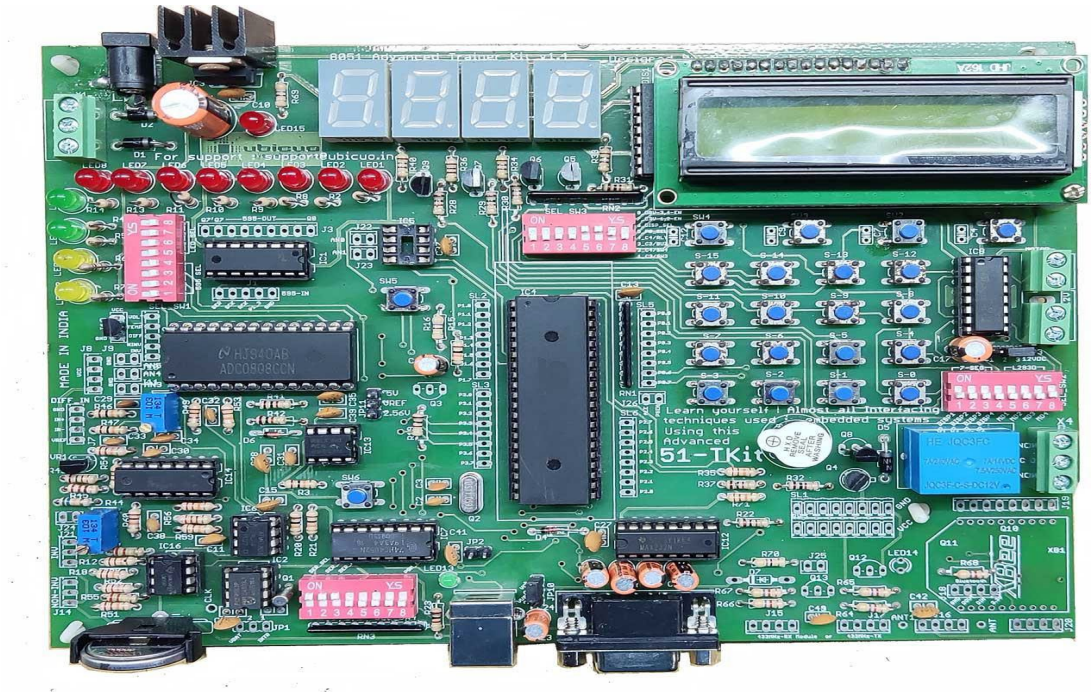
Apparatus Required

- 8051 Trainer Kit
- Power supply (+5V)
- Connecting wires

Theory

The **8051 Trainer Kit** is a development system used to learn programming, interfacing, and operation of the 8051 microcontroller. It contains CPU, memory, I/O ports, timers, and serial communication facilities.

Features to be Verified



1. Microcontroller (CPU)

- Based on 8-bit 8051 architecture
- Contains ALU, registers, control unit
  ✔ Verified by executing simple programs

2. Memory Organization

- **RAM**: Temporary data storage
- **ROM/Flash**: Program storage
  ✔ Verified by storing and retrieving data

3. I/O Ports

- Four 8-bit ports (P0, P1, P2, P3)
  ✔ Verified by:
- Connecting LEDs/switches
- Observing output

4. Timers/Counters

- Timer 0 and Timer 1
  ✔ Verified by generating delays or counting pulses

5. Serial Communication

- UART (TXD, RXD)
  ✔ Verified by communication with PC

6. Interrupt System

- External and internal interrupts
  ✔ Verified using interrupt-based programs

7. Oscillator / Clock Circuit

- Provides clock pulses (typically 11.0592 MHz)
  ✔ Verified by proper execution timing

8. Keyboard / Keypad

- Used for program entry
  ✔ Verified by entering data manually

9. Display Unit

- 7-segment / LCD display
  ✔ Verified by showing:
- Data
- Addresses
- Results

10. Reset Circuit

- Initializes the microcontroller
  ✔ Verified by pressing reset button

11. On-board Peripherals

- ADC/DAC (optional)
- Relay, buzzer, LEDs
  ✔ Verified by interfacing experiments

12. Programming Interface

- Supports assembly/C programming
  ✔ Verified by uploading and executing code

Procedure

1. Switch ON the trainer kit
2. Press RESET
3. Observe display initialization
4. Enter a simple program
5. Execute the program
6. Verify output using LEDs/display
7. Test I/O ports and timers
8. Check serial communication (if available)

Result

All features of the 8051 Trainer Kit were successfully verified, including CPU operation, memory, I/O ports, timers, interrupts, and serial communication.

Precautions

- Ensure proper power supply
- Avoid loose connections
- Enter correct program codes
- Reset before execution

Experiment:05

ALP based on Data transfer Instructions & Arithmetic Instructions(addition) in 8051

Aim

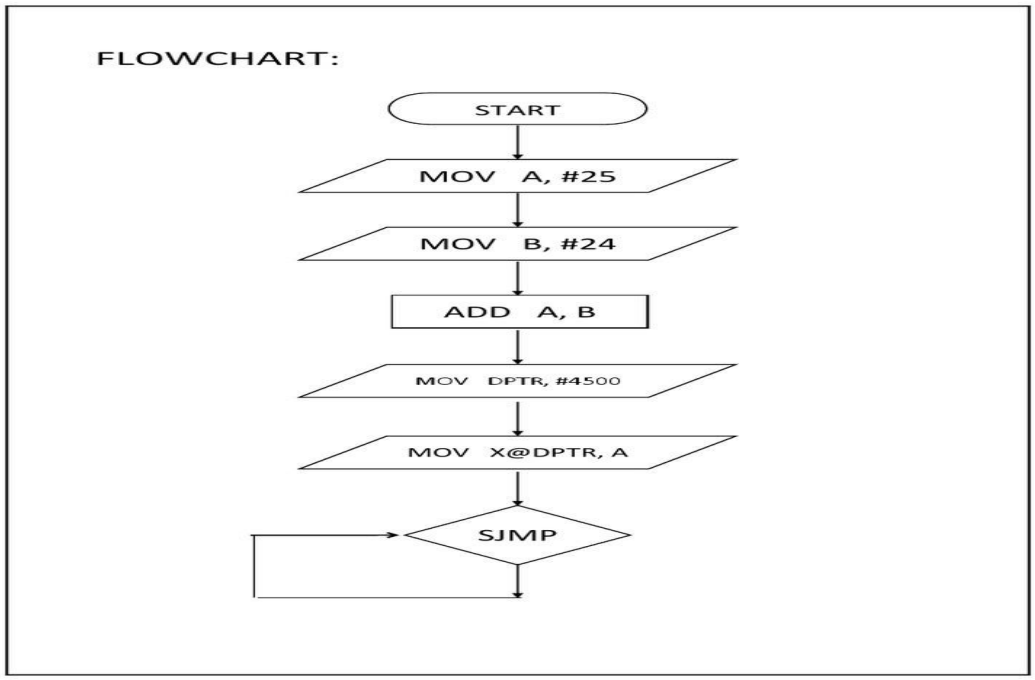
To write and execute an **Assembly Language Program (ALP)** for addition using data transfer and arithmetic instructions in 8051 microcontroller.

Theory

- **Data Transfer Instructions**: MOV, MOVX, MOVC
- **Arithmetic Instructions**: ADD, ADDC, INC

☞ In 8051, arithmetic operations are mainly performed using the **Accumulator (A)**.

Program 1: Addition of Two 8-bit Numbers (Registers)



Algorithm

1. Load first number into accumulator
2. Load second number into another register
3. Add the numbers
4. Store the result

💻 ALP Code

```
MOV A, #05H     ; Load first number into A
MOV R1, #03H    ; Load second number into R1
ADD A, R1       ; A = A + R1
MOV 30H, A      ; Store result at memory location 30H
END
```

📊 Explanation

- `MOV` → Data transfer
- `ADD` → Arithmetic addition
- Result = **08H** stored at **30H**

Program 2: Addition of Two Numbers from Memory

🔷 Algorithm

1. Load first number from memory
2. Move second number
3. Add
4. Store result

💻 ALP Code

```
MOV A, 30H      ; Load first number from 30H
MOV R1, 31H     ; Load second number from 31H
ADD A, R1       ; Add
MOV 32H, A      ; Store result at 32H
END
```

📊 Explanation

- Uses internal RAM locations
- Result stored at **32H**

Program 3: Addition with Carry

💻 ALP Code

```
MOV A, #0FFH     ; Load first number
MOV R1, #02H     ; Load second number
ADD A, R1        ; Add (carry generated)
MOV 30H, A       ; Store result
CLR A            ; Clear A
ADDC A, #00H     ; Add carry to A
MOV 31H, A       ; Store carry
END
```

Result

The ALP was successfully executed and verified using data transfer and arithmetic instructions for addition in 8051.

Precautions

- Use correct memory locations
- Check carry flag for overflow
- Write proper syntax
- Reset before execution

Experiment:06

ALP based on Data transfer Instructions and arithmetic Instructions (subtraction) in 8051

Aim

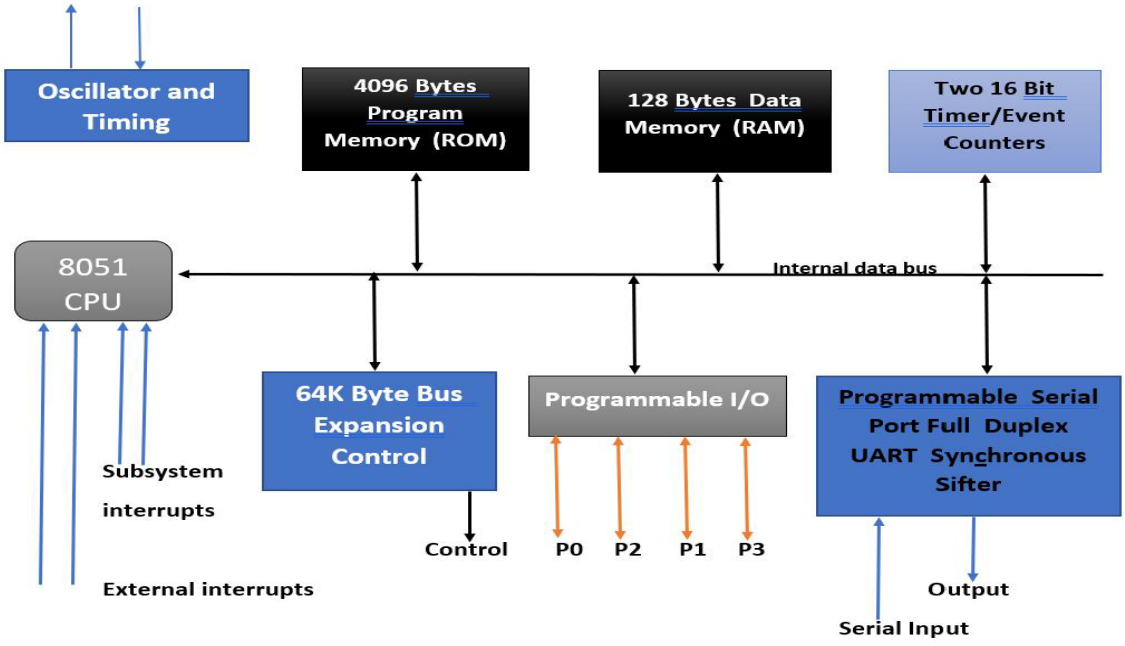
To write and execute an **Assembly Language Program (ALP)** for subtraction using data transfer and arithmetic instructions in 8051 microcontroller.

Theory

- **Data Transfer Instructions**: `MOV`, `MOVX`, `MOVC`
- **Arithmetic Instructions (Subtraction)**: `SUBB` (Subtract with Borrow)

☞ In 8051, subtraction is done using **`SUBB`**, which subtracts **with carry (borrow)**.
☞ So, carry flag must be cleared before subtraction.

Program 1: Subtraction of Two 8-bit Numbers (Registers)



◈ Algorithm

1. Load first number into accumulator
2. Load second number into register
3. Clear carry
4. Subtract
5. Store result

💻 ALP Code

```
MOV A, #09H     ; Load first number
MOV R1, #04H    ; Load second number
CLR C           ; Clear carry (important)
SUBB A, R1      ; A = A - R1
MOV 30H, A      ; Store result
END
```

📊 Explanation

- `MOV` → Data transfer
- `SUBB` → Arithmetic subtraction
- `CLR C` → ensures no previous borrow
- Result = **05H**

Program 2: Subtraction of Two Numbers from Memory

◈ Algorithm

1. Load first number
2. Load second number
3. Clear carry
4. Subtract
5. Store result

💻 ALP Code

```
MOV A, 30H      ; Load first number
MOV R1, 31H     ; Load second number
CLR C           ; Clear carry
SUBB A, R1      ; Subtract
```

```
MOV 32H, A      ; Store result
END
```

📊 Explanation

- Uses internal RAM locations
- Result stored at **32H**

Program 3: Subtraction with Borrow

💻 ALP Code

```
MOV A, #05H     ; Load first number
MOV R1, #09H    ; Load second number
CLR C           ; Clear carry
SUBB A, R1      ; Subtract (borrow occurs)
MOV 30H, A      ; Store result
MOV A, #00H     ; Clear A
ADDC A, #00H    ; Move carry (borrow) to A
MOV 31H, A      ; Store borrow
END
```

Result

The subtraction operation was successfully performed using data transfer and arithmetic instructions in 8051.

Precautions

- Always use `CLR C` before `SUBB`
- Check carry flag for borrow
- Use correct memory addresses
- Ensure proper syntax

Experiment:07

Write an ALP to generate delay using Timer

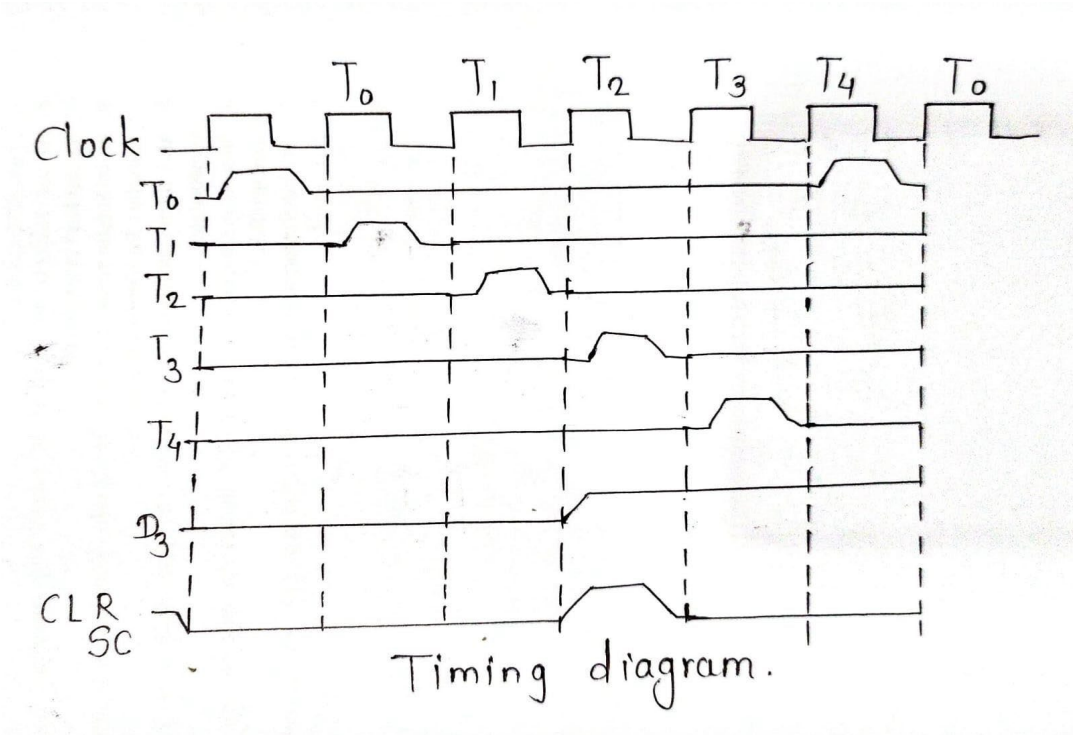
Aim

To write an **Assembly Language Program (ALP)** to generate a time delay using the timer of 8051 microcontroller.

Theory

- 8051 has **Timer 0 and Timer 1**
- Timers can operate in different modes (Mode 0, 1, 2, 3)
- **Mode 1 (16-bit timer)** is commonly used for delay generation
- Timer counts machine cycles and overflows after reaching maximum value

Program: Delay using Timer 0 (Mode 1)



Timing diagram.

◈ Algorithm

1. Load TMOD register to select Timer mode
2. Load initial values in TH0 and TL0
3. Start timer (TR0 = 1)
4. Wait until overflow (TF0 = 1)
5. Stop timer and clear flag
6. Repeat if needed

💻 ALP Code

```
ORG 0000H        ; Start address

MOV TMOD, #01H   ; Timer 0, Mode 1 (16-bit)

BACK: MOV TH0, #0FCH   ; Load high byte
      MOV TL0, #018H   ; Load low byte

      SETB TR0         ; Start Timer 0

WAIT: JNB TF0, WAIT    ; Wait until TF0 = 1 (overflow)

      CLR TR0          ; Stop timer
      CLR TF0          ; Clear overflow flag

      SJMP BACK        ; Repeat delay

END
```

Explanation

- `TMOD = 01H` → Timer 0 Mode 1
- `TH0`, `TL0` → Initial count values
- `TR0` → Starts timer
- `TF0` → Overflow flag
- Loop creates continuous delay

Delay Calculation (Example)

For 11.0592 MHz crystal:

- Machine cycle = 1.085 µs
- Count = 65536 – Initial Value

👉 Delay ≈ Count × Machine Cycle

Result

The delay was successfully generated using Timer 0 of 8051.

Precautions

- Load correct initial values
- Clear TF0 before reuse
- Use proper timer mode

Experiment:08

Interfacing 7 segment display with 8051

Aim

To interface a **7-segment display** with the 8051 microcontroller and display digits (0–9).

Components Required

- 8051 Trainer Kit
- 7-Segment Display (Common Anode/Cathode)
- Resistors (220Ω–330Ω)
- Connecting wires

Theory

A **7-segment display** consists of 7 LEDs (a–g) used to display digits.

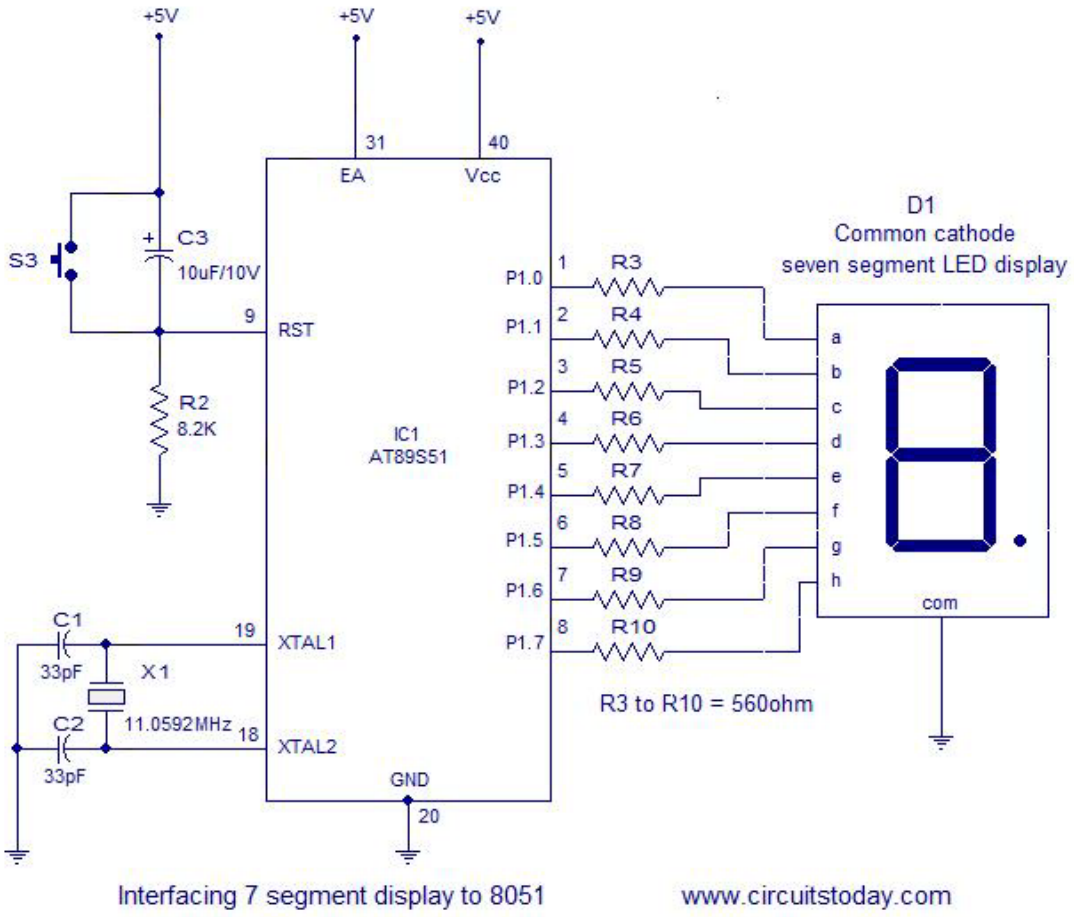
👉 Two types:

- **Common Cathode (CC)** → Common pin connected to GND
- **Common Anode (CA)** → Common pin connected to Vcc

👉 8051 sends binary data to segments via I/O ports (usually **Port 1 or Port 2**).

Circuit Connection (Concept)

- Connect segments **a–g** to Port 1 pins (P1.0 – P1.6)
- Use resistors in series
- Common pin:
  - CC → GND
  - CA → +5V



Segment Code Table (Common Cathode)

Digit	Hex Code
0	3FH
1	06H
2	5BH
3	4FH
4	66H
5	6DH
6	7DH
7	07H
8	7FH
9	6FH

Program: Display Digits (0–9)

💻 ALP Code

```
ORG 0000H

MOV P1, #00H     ; Initialize port

START:
MOV P1, #3FH     ; Display 0
ACALL DELAY

MOV P1, #06H     ; Display 1
ACALL DELAY

MOV P1, #5BH     ; Display 2
ACALL DELAY

MOV P1, #4FH     ; Display 3
ACALL DELAY

MOV P1, #66H     ; Display 4
ACALL DELAY

MOV P1, #6DH     ; Display 5
ACALL DELAY

MOV P1, #7DH     ; Display 6
ACALL DELAY

MOV P1, #07H     ; Display 7
ACALL DELAY

MOV P1, #7FH     ; Display 8
ACALL DELAY

MOV P1, #6FH     ; Display 9
ACALL DELAY

SJMP START       ; Repeat

; Delay Subroutine
DELAY:
MOV R2, #255
D1: MOV R3, #255
D2: DJNZ R3, D2
    DJNZ R2, D1
RET

END
```

Explanation

- `MOV P1, value` → Sends data to 7-segment
- Each hex value represents a digit
- Delay is used to make digits visible
- Loop displays digits continuously

Result

Digits **0 to 9** were successfully displayed on the 7-segment display using 8051.

Precautions

- Check display type (CC or CA)
- Use proper resistors
- Ensure correct wiring
- Avoid short circuit