

# 8-Bit Arithmetic Operations Using 8051

## Aim:

To perform 8-bit arithmetic operations such as addition, subtraction, multiplication, and division using the 8051 microcontroller.

## Apparatus Required:

Laptop with Keil uVision software

## Algorithm:

### 1. Addition

1. Start the program.
2. Load the first 8-bit number from memory location **30H** into register **A**.
3. Load the second 8-bit number from memory location **31H** into register **B**.
4. Add the contents of **A** and **B**.
5. Store the result (sum) in memory location **40H**.
6. Check if a carry is generated.
7. If carry = 1, store carry in memory location **41H**, else store 00H.

### 2. Subtraction

1. Load the first number (minuend) from **30H** into register **A**.
2. Load the second number (subtrahend) from **31H** into register **B**.
3. Clear the carry flag.
4. Subtract **B** from **A** using **SUBB**.
5. Store the result in memory location **40H**.

### 3. Multiplication

1. Load the first number from **30H** into register **A**.
2. Load the second number from **31H** into register **B**.
3. Multiply **A**  $\times$  **B** using the **MUL AB** instruction.

4. Store the **lower byte** (present in A) in memory location **40H**.
5. Store the **higher byte** (present in B) in memory location **41H**.

## 4. Division

1. Load the dividend from **30H** into register **A**.
2. Load the divisor from **31H** into register **B**.
3. Perform division using **DIV AB**.  
Store the **quotient** (present in A) in memory location **40H**.
4. Store the **remainder** (present in B) in memory location **41H**.

## 5. Stop

1. End the program execution by jumping to itself (infinite loop).

### For Addition:

1. Load the first number from memory location 30H into register A.
2. Load the second number from memory location 31H into register B.
3. Add the contents of registers A and B.
4. Store the result in memory location 40H.
5. Store the carry (if any) in 41H.

### For Subtraction:

1. Load the first number from memory location 30H into register A.
2. Load the second number from memory location 31H into register B.
3. Subtract B from A.
4. Store the result in memory location 40H.

### For Multiplication:

1. Load the first number from memory location 30H into register A.
2. Load the second number from memory location 31H into register B.
3. Multiply A and B.
4. Store the lower byte of the result in memory location 40H.

5. Store the higher byte of the result in memory location 41H.

## For Division:

1. Load the dividend from memory location 30H into register A.
2. Load the divisor from memory location 31H into register B.
3. Divide A by B.
4. Store the quotient in memory location 40H.
5. Store the remainder in memory location 41H.

## Programs:

```
ORG 0000H      ; Start program from address 0000H
LJMP MAIN      ; Jump to the main routine
```

### MAIN:

```
; --- Set up common data for all operations ---
MOV R1, #30H    ; First operand
MOV R2, #20H    ; Second operand
; Note: For this combined code, we use registers R1 and R2 for inputs.

; --- ADDITION (Result in R3, Carry in R4) ---
MOV A, R1      ; Load R1 into Accumulator A
ADD A, R2      ; Add R2 to A
MOV R3, A      ; Store sum in R3 (Result: 50H)
; Check carry manually if needed, for simplicity we skip storing carry here in the combined
code.

; --- SUBTRACTION (Result in R5) ---
MOV A, R1      ; Load R1 (minuend) into A
CLR C          ; Clear the Carry flag (essential for SUBB to work correctly)
SUBB A, R2     ; Subtract R2 (subtrahend) from A with borrow
MOV R5, A      ; Store difference in R5 (Result: 10H)

; --- MULTIPLICATION (Low byte in R6, High byte in B/R7) ---
MOV A, R1      ; Load first operand into A
MOV B, R2      ; Load second operand into B
MUL AB        ; Multiply A and B (Product: A=Low byte, B=High byte)
MOV R6, A      ; Store low byte in R6 (Result: 60H, as 30h * 20h = 600h)
; The high byte is already in B register, you could move B to R7 if desired: MOV R7, B

; --- DIVISION (Quotient in A, Remainder in B) ---
MOV A, R1      ; Load dividend into A
MOV B, R2      ; Load divisor into B
DIV AB        ; Divide A by B (Quotient in A, Remainder in B)
```

```
MOV R7, A      ; Store quotient in R7 (Result: 01H, integer division)
; The remainder is in B, you could store it in R0 if desired: MOV R0, B
```

STOP:

```
SJMP STOP      ; Infinite loop to halt the program
```

END

## Output:

The screenshot shows the Keil uVision IDE interface with the following details:

- File Menu:** File, Edit, View, Project, Flash, Debug, Peripherals, Tools, SVCS, Window, Help.
- Toolbar:** Includes icons for Open, Save, Build, Run, Stop, and various simulation and memory tools.
- Registers Window:** Shows the state of registers R0 through R7 and other system variables like SP, PC, and flags. All values are initialized to 0x00 except for R0 which is 0x01.
- Disassembly Window:** Displays the assembly code for EXP6.asm. The code includes the main routine (MAIN) which performs addition, subtraction, multiplication, and division operations on R1 and R2, and a STOP loop at the end.
- Call Stack + Locals Window:** Shows the current call stack and local variable information, with EXP6 listed as the active function.
- Command Line:** Displays the command line interface with the current project path and build status.
- Status Bar:** Shows the current simulation time (t1: 0.00002850 sec), clock (L37 C1), and memory usage (CAP NUM SCRL OVR R/W).

The results of addition, subtraction, multiplication, and division operations will be stored in memory locations 40H and 41H as specified in the program.

## Result:

The 8-bit arithmetic operations using the 8051 microcontroller have been successfully executed and verified using Keil software.