SNS COLLEGE OF TECHNOLOGY
COIMBATORE - 641035

MICROPROCESSORS AND MICROCONTROLLERS

CASE STUDY : 8051 ASSEMBLY LANGUAGE

WHAT IS 8051 ASSEMBLY LANGUAGE PROGRAMMING

- The 8051 assembly language programming is based on the memory registers. If we want to manipulate data to a processor or controller by performing subtraction, addition, etc., we cannot do that directly in the memory, but it needs registers to process and to store the data.
- Assembly language is a low level programming language used to write program code in terms of mnemonics

- The assembly language mnemonics are in the form of op-code, such as MOV, ADD, JMP, and so on, which are used to perform the operations
- Op-code: The op-code is a single instruction that can be executed by the CPU. Here the op-code is a MOV instruction.
- Operands: The operands are a single piece of data that can be operated by the op-code. Example, multiplication operation is performed by the operands that are multiplied by the operand.

Registers: The 8051 has four register banks, each containing eight registers (R0-R7). Registers R0 and R1 are used as the accumulator (A) and the B register, respectively.Instructions: The instructions in 8051 assembly language are typically one-byte long and can be categorized into various types such as data transfer, arithmetic, logical, branch, and control instructions.Data Transfer Instructions:MOV: Moves data from one register/memory location to another.Example: MOV A, #25H (Move immediate value 25H to accumulator A)

XCH: Exchanges the content of a register/memory location with the accumulator.Example: XCH A, R1 (Exchange accumulator A with register R1)Arithmetic Instructions:ADD: Adds the accumulator with another register/memory location.Example: ADD A, R2 (Add the content of register R2 to accumulator A)SUBB: Subtracts the content of another register/memory location from the accumulator with borrow.Example: SUBB A, @R0 (Subtract the content of the address pointed by R0 from A with borrow)

Programming in 8-bit 8051 assembly language involves writing code that operates on 8-bit data and instructions.

Registers and Data Size: The 8051 microcontroller has a set of registers, including the accumulator (A) and general-purpose registers (R0-R7), each capable of holding 8 bits of data. This means that data manipulation operations, such as addition, subtraction, logical AND/OR, are performed on 8-bit data at a time.Memory Architecture: The 8051 has an addressable memory space of 64 KB. Each memory location holds 8 bits (1 byte) of data. The memory is organized into different banks and areas, such as code memory (program memory), data memory, and special function registers (SFRs)

Example:

```
ORG 0H          ; Start of program memory

MOV A, #25H   ; Load accumulator A with the first number (e.g., 25H)
MOV B, #3AH   ; Load register B with the second number (e.g., 3AH)
ADD A, B    ; Add the contents of accumulator A and register B

MOV 30H, A   ; Store the result in memory location 30H

END ; End of program
```

- ORG 0H: This directive sets the origin of the program to memory address 0.
- MOV A, #25H: This instruction loads accumulator A with the first 8-bit number (e.g., 25H).
- MOV B, #3AH: This instruction loads register B with the second 8-bit number (e.g., 3AH).
- ADD A, B: This instruction adds the contents of accumulator A and register B and stores the result in accumulator A.
- MOV 30H, A: This instruction stores the result (sum) in memory location 30H.

- ORG 0H ; Start of program memoryMAIN:
- MOV A, #HIGH SOURCE ; Load high byte of source number into accumulator
- MOV B, #HIGH DEST ; Load high byte of destination number into B register
- SUBB A, B ; Subtract the high bytes, with borrow from previous operation
- MOV DEST_H, A ; Store the result in the high byte of the destination
- MOV A, #LOW SOURCE ; Load low byte of source number into accumulator
- MOV B, #LOW DEST ; Load low byte of destination number into B register

- SUBB A, B ; Subtract the low bytes, with borrow from previous operation
- MOV DEST_L, A ; Store the result in the low byte of the destination
- END ; End of programSOURCE:
- DW 1234H ; Source number (16-bit)DEST:
- DW 5678H ; Destination number (16-bit)DEST_H:
- DS 1 ; Destination high byteDEST_L:
- DS 1 ; Destination low byte

- MOV A, #HIGH SOURCE: Load the high byte of the source number into the accumulator.
- MOV B, #HIGH DEST: Load the high byte of the destination number into register B.
- SUBB A, B: Subtract the high bytes, with borrow from the previous operation. This handles any carry from the subtraction of the low bytes.
- MOV DEST_H, A: Store the result of the subtraction in the high byte of the destination.Repeat the above steps for the low bytes.
- END: Marks the end of the program

THANK YOU