Lab 02

Tuesday 5th November, 2024

Lab Objective

This lab will teach you how to perform basic data movement operations using the 8086 assembly language. By the end of this session, you will understand how to transfer data between registers, from memory to registers, and vice versa.

1. Introduction to Data Movement Instructions

The 8086 microprocessor has a variety of instructions for moving data between registers and memory locations. Some of the most common data movement instructions include:

- **MOV**: Transfers data from one location to another.
- **XCHG**: Exchanges the contents of two registers or a register and a memory location.
- **PUSH/POP**: Used for stack operations to save and restore data.
- **LEA**: Loads the effective address of a variable into a register.

2. Commonly Used Instructions

MOV Instruction

Syntax: `MOV destination, source`

Moves data from the `source` to the `destination`. The `source` and `destination` can be registers, memory locations, or immediate values.

Example:

```
MOV AX, BX      ; Copy the contents of BX into AX
MOV CX, 1234h   ; Load the immediate value 1234h into CX
MOV [5000h], AX ; Move the value in AX to memory address 5000h
```

XCHG Instruction

Syntax: `XCHG operand1, operand2`

Exchanges the values between `operand1` and `operand2`.

Example:

```
XCHG AX, BX     ; Exchange the contents of AX and BX
```

PUSH/POP Instructions

PUSH: Places a word on the stack.
POP: Removes a word from the stack.

Example:

```
PUSH AX         ; Save AX onto the stack
POP BX          ; Restore the saved value into BX
```

LEA Instruction

Syntax: `LEA register, source`
Loads the address of the `source` into the `register`.
Example:

```
LEA SI, [5000h] ; Load the address 5000h into SI
```

3. Lab Exercises

Exercise 1: Basic Register-to-Register Data Movement

1. Write a program to move data between various registers.

Exercise 2: Moving Data from Memory to Registers

1. Write a program to load data from a specific memory address into a register and vice versa.

Exercise 3: Stack Operations

1. Write a program to demonstrate the use of the `PUSH` and `POP` instructions.