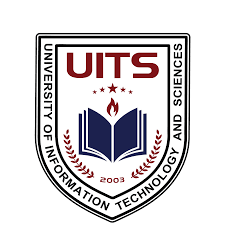
University of Information Technology & Sciences

Department of

Computer Science and Engineering



**Lab Report-01**

Course Title: Microprocessors and Microcontrollers Lab

Course Code: CSE-360

Submitted To

Md. Ismail

Lecturer

Of

CSE Department

Submitted By

Name : Mobarok Hossain Zobaer

Id : 0432220005101029

Batch : 52

Semester: Spring 25

Section : 6A1

**Problem Description: Different Types of Registers in a Microprocessor**

1. General-purpose Registers (GPRs):

* AX (Accumulator)
* BX (Base)
* CX (Count)
* DX (Data)

Applications:

1. Storing intermediate results of arithmetic or logical operations.
2. Holding data for quick access during program execution.
3. Special-purpose Registers:

* Program Counter (PC)
* Stack Pointer (SP)
* Flags Register (FR) / Status Register

Applications:

1. Program flow control (Program Counter, Stack Pointer).
2. Handling program status (Flags Register).
3. Index Registers:

* SI (Source Index)
* DI (Destination Index)

Applications:

1. Array and string manipulation.
2. Memory block transfers.
3. Control Registers:

* Control Register 0 (CR0)
* Control Register 3 (CR3)

Applications:

1. System-level control such as enabling/disabling features (e.g., paging in virtual memory).
2. Managing memory access.
3. Segment Registers:

* CS (Code Segment)
* DS (Data Segment)
* SS (Stack Segment)
* ES (Extra Segment)

Applications:

1. Memory segmentation for organized access to different memory regions.
2. Fundamental Operation Codes (Opcodes) for Assembly Language

**Problem Description: Fundamental opcodes in assembly language**

1. MOV (Move):

Syntax: MOV destination, source

Function: Copies data from the source operand to the destination operand.

Example: MOV AX, 10 (Moves the value 10 into register AX).

1. ADD (Addition):

Syntax: ADD destination, source

Function: Adds the source operand to the destination operand and stores the result in the destination.

Example: ADD AX, BX (Adds the value in BX to AX).

1. SUB (Subtraction):

Syntax: SUB destination, source

Function: Subtracts the source operand from the destination operand.

Example: SUB AX, BX (Subtracts the value in BX from AX).

1. MUL (Multiply):

Syntax: MUL operand

Function: Multiplies the accumulator (AX) by the operand (usually the value in a register or memory).

Example: MUL BX (Multiplies AX by BX).

1. DIV (Divide):

Syntax: DIV operand

Function: Divides the accumulator (AX) by the operand (usually a register or memory value).

Example: DIV BX (Divides AX by BX).

1. CMP (Compare):

Syntax: CMP operand1, operand2

Function: Compares two operands by subtracting the second from the first but does not store the result. It sets the flags in the status register based on the comparison.

Example: CMP AX, BX (Compares AX with BX).

1. JMP (Jump):

Syntax: JMP label

Function: Unconditionally jumps to the specified label or address.

Example: JMP START (Jumps to the instruction at the label START).

1. JE / JZ (Jump if Equal / Jump if Zero):

Syntax: JE label or JZ label

Function: Jumps to the specified label if the Zero flag (Z) is set (indicating equality or zero result).

Example: JE EQUAL (Jumps to the EQUAL label if the result of the last operation was zero).

1. CALL (Call Procedure):

Syntax: CALL procedure\_address

Function: Calls a subroutine or function at the specified address.

Example: CALL PRINT\_MSG (Calls the subroutine PRINT\_MSG).

1. RET (Return from Subroutine):

Syntax: RET

Function: Returns from a subroutine and passes control back to the calling function.

Example: RET (Returns control to the calling function).

1. PUSH (Push onto Stack):

Syntax: PUSH operand

Function: Pushes the operand (a register or value) onto the stack.

Example: PUSH AX (Pushes the value of AX onto the stack).

1. POP (Pop from Stack):

Syntax: POP operand

Function: Pops the top value from the stack into the operand (usually a register).

Example: POP AX (Pops the top value from the stack into AX).