**COMPUTER ORGANIZATION & ASEMBLY LANGUAGE**

**LAB TASK 02**

**ALIZA RIZVI**

**58782**

1. Describe the function of each:

**Accumulator Register (A, AX, EAX, RAX)**

* Used for arithmetic and logical operations.
* Many instructions use this register by default (example: multiplication, division)

**Base Register (B, BX, EBX, RBX)**

* Often used as a base pointer for memory access.
* Can hold an address or offset while working with arrays or memory.

**Count Register (C, CX, ECX, RCX)**

* Used as a counter in loops and string operations.

**Data Register (D, DX, EDX, RDX)**

* Used in multiplication, division, and I/O operations.
* In multiplication and division stores high order results**.**

1. Describe the role of index registers in source and destination operations. Provide an example using SI and DI in an assembly instruction

* Index registers (SI and DI) help with copying or moving data in memory.
  + Source Index (SI): Holds the address of the source data.
  + Destination Index (DI): Holds the address where data should be moved.
  + Used in string operations like copying arrays.
* Examples:

**MOV SI, 1000h**: Source address

**MOV DI, 2000h**: Destination address

**MOV CX, 10**: Counter for 10 elements

**CLD:** Clear Direction Flag (move forward)

**REP MOVSB**: Move bytes from [SI] to [DI], repeat for CX times

1. Briefly describe the functions of these bits:

* Overflow Flag (OF)
* Zero Flag (ZF)
* Carry Flag (CF)

**Overflow Flag (OF)**

• Set when an arithmetic operation exceeds the max value a register can hold.

• Example: Adding two large positive numbers gives a negative result due to overflow.

**Zero Flag (ZF)**

• Set when a result is zero.

• Example: SUB AL, AL makes AL = 0, so ZF = 1.

**Carry Flag (CF)**

• Set when an arithmetic operation results in a carry (for addition) or borrow (for subtraction).

• Example: ADD AL, 255 when AL = 1 causes CF to be set because 256 cannot fit in 8 bits.

1. Write a simple program using MOV, ADD, and SUB instructions:

* Move values to registers
* Perform addition and subtraction
* Store and display result
* MOV AL, 10: Move 10 into AL

MOV BL, 5: Move 5 into BL

* ADD AL, BL: AL = AL + BL (10 + 5 = 15)

SUB AL, 2: AL = AL - 2 (15 - 2 = 13)

; Exit program

MOV RAX, 60: syscall: exit

MOV RDI, 0: return 0

syscall

• AL = 10, BL = 5

• AL = AL + BL →15

• AL = AL - 2 → 13

• Program exits with result 13 stored in AL.