8086 Instruction Set

data movement
arithmetic
logic operations
Control flow instruction
I/P / O/P instruction
String

data movement

MOV — Transfer data from Source to destination

XCHG — Swap contents of 2 registers or register & memory location

PUSH — push data on stack

POP — pop data from Stack

LEA — Load Effective address loads addr of memory operand into register

Arithmetic

ADD, SUB, — Sum, difference,
ADC, SBB — Sum with Carry, difference with borrow

INC DEC — Increment, decrement

MUL, IMUL, DIV, IDIV — multiplication & division (unsigned, signed)

Logic

AND, OR, XOR, NOT — Bitwise logical operations

TEST — Bitwise AND operand modifies flags while operands remain unchanged

SHL, SHR, SAL, SAR — Shift left & Shift right

ROL, ROR, RCL, RCR — left Rotate, right Rotate

control

Jmp — Jump to specified instruction

JC, JNC, JZ, JNZ, JS, JNS, JO, JNO — Conditional branching on flags

CAL — Call to subroutine

RET — Return from subroutine

INT — S/w interrupt

IRET — interrupt return

String

MOVSB, MOVSW — Transfer byte or word from source to destination

CMPSB, CMPSW — compare byte or string operand

SCASB, SCASW — Scan byte or word string operands

LODSB, LODSW — Get byte or word from memory to accumulator

I/P - O/P

IN — Input from port

OUT — Output to port

Flag Ctrl

CLC, STC, CMC — Set, clear complement flags on carry

CLD, STD — Set & clear direction flag

CLI, STI — Set or clear Interrupt flag

HLT — halt processor execution

Other Instruction

NOP — NO Operation

WAIT — Await external event

ESC — Jump to external co processor

Process Ctrl Instruction

control order execution in a program & in processes.

Branching, looping, calling fn or Subroutines

→ changes seq of instruction execution

changes program flow

Branching Instn

Transfer flow of execution onto certain condition or unconditional transfer to part of program

looping

used to repeat execution of block or code either conditional or unconditionally

Subroutine

Call & return from Subroutine

Unconditional Jump — instruction unconditionally jumps to predetermined location w/o

Conditional Jump

— moves control to specific address depending on the value of condition

Sub routine Calls

— transferring control to subroutine enabling faster execution