Conditional JMP in 8086

Conditional JMP instructions in 8086 are jump instructions that transfer program control to a target address only if a specific condition is met. The condition is determined by the state of flags in the Flag Register.

Key Characteristics:

1. Always use CMP before conditional jumps
2. Conditional jumps test flag conditions (ZF, CF, SF, OF, PF)
3. They are short jumps (displacement of -128 to +127 bytes from current IP)
4. Used for implementing decision-making and loops

What is a Conditional Jump?

A conditional jump instruction checks a condition and:

1. Jumps to a label/address if the condition is satisfied
2. Continues sequential execution if the condition is not satisfied

Common Conditional JMP Instructions

Instruction	Condition	Flag Test
JE/JZ	Jump if equal / zero	ZF = 1
JNE/JNZ	Jump if not equal / not zero	ZF = 0
JG/JNLE	Jump if greater (signed)	ZF=0 and SF=OF
JL/JNGE	Jump if less (signed)	SF ≠ OF
JA/JNBE	Jump if above (unsigned)	CF=0 and ZF=0
JB/JNAE	Jump if below (unsigned)	CF=1
JC	Jump if carry	CF=1
JNC	Jump if no carry	CF=0
JO	Jump if overflow	OF=1
JNO	Jump if no overflow	OF=0
JS	Jump if sign (negative)	SF=1
JNS	Jump if not sign (positive)	SF=0
JP/JPE	Jump if parity even	PF=1
JNP/JPO	Jump if parity odd	PF=0

CMP Instruction in 8086 Assembly: Complete Definition & Details

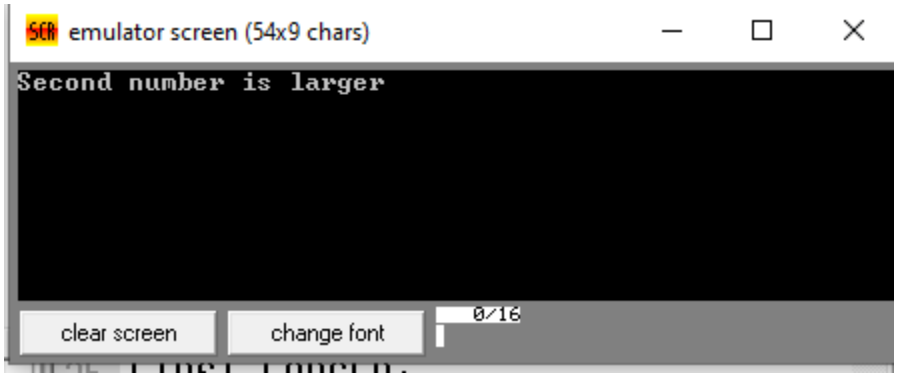
The CMP (Compare) instruction is one of the most fundamental and frequently used instructions in 8086 assembly programming. It performs a comparison between two operands by subtracting the second operand from the first, updates the CPU flags based on the result, but discards the result (doesn't store it anywhere).

Noted: CMP subtracts the source operand from the destination operand:

Example-1: Write Assembly code to find larger number between into two numbers.

```
.MODEL SMALL
.STACK 100H
.DATA
   num1 DB 25
   num2 DB 42
   msg1 DB 'First number is larger$'
   msg2 DB 'Second number is larger$'
   msg3 DB 'Both are equal$'

.CODE
MAIN PROC
   MOV AX, @DATA
   MOV DS, AX

   MOV AL, num1
   CMP AL, num2      ; Compare num1 with num2

   JG FIRST_LARGER    ; Jump if num1 > num2
   JL SECOND_LARGER   ; Jump if num1 < num2
   ; If equal
   MOV DX, OFFSET msg3
   JMP DISPLAY

FIRST_LARGER:
   MOV DX, OFFSET msg1
   JMP DISPLAY

SECOND_LARGER:
   MOV DX, OFFSET msg2

DISPLAY:
   MOV AH, 09H
   INT 21H

   MOV AH, 4CH
   INT 21H
MAIN ENDP
END MAIN
```

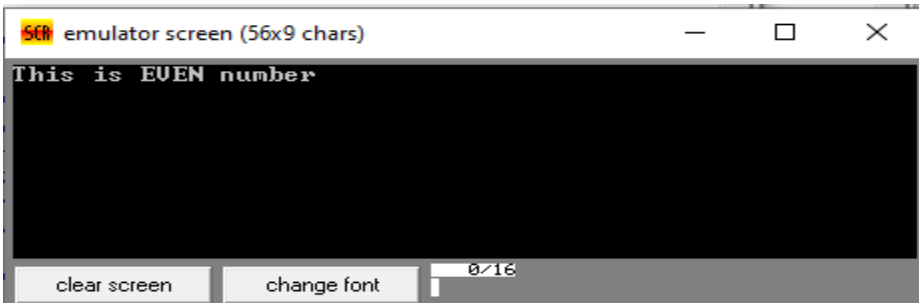
OUTPUT



Example-2: Write Assembly code to find EVEN or ODD number.

```
.MODEL SMALL
.STACK 100H
.DATA
   num1 DB 22

   msg1 DB 'This is EVEN number $'
   msg2 DB 'This is ODD  number $'
.CODE
MAIN PROC
   MOV AX, @DATA
   MOV DS, AX

   MOV AX,0000

   MOV AL, num1
   MOV BL,2
   DIV BL
  CMP AH,0
   JNE DISPLAYODD

   MOV AH,09H
   MOV DX,OFFSET msg1
   INT 21H
   JMP EXITPRG

 DISPLAYODD:
   MOV AH,09H
   MOV DX,OFFSET msg2
   INT 21H
 EXITPRG:
   MOV AH,4CH
   INT 21H

MAIN ENDP
END MAIN
```

OUTPUT



TEST Command in 8086 Microprocessor

The TEST instruction performs a logical AND operation between two operands but does not store the result. It only updates the flag registers (SF, ZF, PF) based on the result.

Syntax

TEST destination, source

Operation

destination AND source → result (not stored)
Update flags based on result

Key Points

1. Operands remain unchanged
2. Affects flags: SF, ZF, PF (also AF, CF, OF are cleared)
3. Commonly used for bit testing without modifying data

Flag Effects

Flag	Effect
SF	Set to MSB of result
ZF	Set if result is zero
PF	Set if low byte has even number of 1 bits
CF, OF, AF	Always cleared to 0

Common Applications

1. Bit testing in device status registers
2. Checking parity of data
3. Zero detection without altering data
4. Condition checking for loops and branches

The ***TEST*** instruction is particularly useful in real-time systems and device driver programming where you need to check hardware status bits without modifying them.

```
.MODEL SMALL
.STACK 100H
.DATA
   NUM  DB 55      ; Number to test (37h )
   MSG1 DB "This Number is even$"
   MSG2 DB "This Number is odd$"
.CODE
MAIN PROC
   MOV AX, @DATA
   MOV DS, AX

   MOV AL, NUM     ; Load number
   TEST AL, 01H    ; Test least significant bit
   JZ EVEN_NUM     ; If ZF=1 (LSB=0), number is even
```

```
; Odd number
   LEA DX, MSG2
   JMP DISPLAY

EVEN_NUM:
   LEA DX, MSG1

DISPLAY:
   MOV AH, 09H     ; DOS display string function
   INT 21H

   MOV AH, 4CH     ; Exit program
   INT 21H

MAIN ENDP
END MAIN
```

Exercise

Theory Questions.

Practical Questions.

Objective and MCQs: