

## Loops in 8086 Assembly Language

A loop in 8086 assembly is a programming construct that executes a sequence of instructions multiple times. The 8086 processor provides several methods to implement loops, with the most common being the LOOP instruction. The LOOP instruction depend on CX register as a counter. The loop instruction control transfer always up side where write specific label or paragraph name. In the 8086 processor, the CX register usually controls looping. The processor decreases the value of CX automatically each time the loop runs.

### Syntax:

LOOP label\_Name:

### How it works:

1. CX is automatically used by loop instructions
2. Decrements CX by 1 automatically
3. If  $CX \neq 0$ , jumps to the specified label or paragraph name
4. If  $CX = 0$ , continues to the next instruction
5. Maximum loop count: 65535 (FFFFH)

### Example 1: Basic LOOP (Print A Character 5 times)

```
.MODEL SMALL
.STACK 100H
.DATA
.CODE
MAIN PROC

    MOV CX, 5           ; Set counter to 5

LOOP_START:
    MOV AH, 02h        ; 02h DOS function/service for print one character to 'A'
    MOV DL,'A'
    INT 21H           ; Execute 02h function /service of Print one ASCII character
    LOOP LOOP_START  ; Decrement CX and loop if not zero

    MOV AH, 4CH       ; Exit program
    INT 21H
MAIN ENDP
```



**Example 2:** Basic LOOP (Print " Pakistan " String 8 times)

```

.MODEL SMALL
.STACK 100H
.DATA
    Text1 dw " Paskistan $"

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

    MOV CX, 8          ; Set counter to 8

Print_String:
    MOV AH, 9h        ; 09h DOS function /service to print string
    MOV DX, offset Text1
    INT 21H           ; Execute 09h dos function /service
    LOOP Print_String ; Decrement CX and loop if not zero

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

The screenshot shows a terminal window titled "emulator screen (80x9 chars)". The output on the screen is the word "Paskistan" repeated eight times. Below the terminal area, there are two buttons: "clear screen" and "change font", along with a font size indicator set to "0/16".

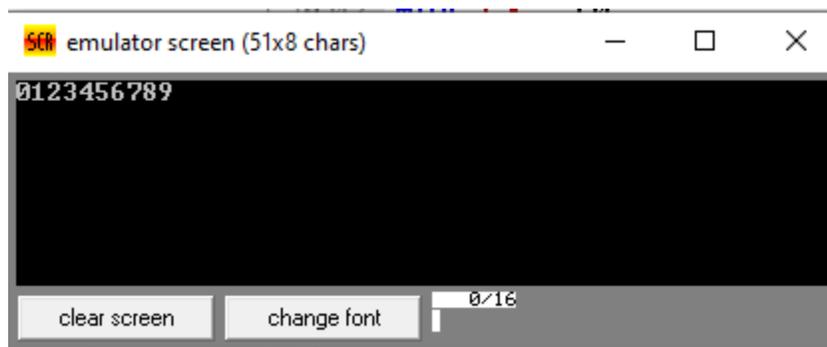
**Example 3:** Basic LOOP ( Print Linear series from 0 to 9 numbers)

```

.MODEL SMALL
.STACK 100H
.DATA
.CODE
MAIN PROC

    MOV CX, 10           ; Set counter to 10
    MOV BL, 0
Linear_Series:
    MOV AH, 2h          ; 02h DOS function to print A character
    MOV DL, BL
    ADD DL, 30H         ; add 30h value in DL register.
    INT 21H             ; Execute 02h function /service of Print one ASCII character
    INC BL              ; Increment 1 by 1 in BL register
    LOOP Linear_Series  ; Decrement CX and loop if not zero

    MOV AH, 4CH         ; Exit program
    INT 21H
MAIN ENDP
    
```



### Loop Instructions Family

The 8086 provides several loop instructions:

Instruction	Description	Condition
LOOP	Decrement CX and loop if CX ≠ 0	CX ≠ 0
LOOPE/LOOPZ	Loop if CX ≠ 0 and ZF = 1	CX ≠ 0 AND ZF = 1
LOOPNE/LOOPNZ	Loop if CX ≠ 0 and ZF = 0	CX ≠ 0 AND ZF = 0

## Summary

1. Loops in 8086 assembly are fundamental for repetitive tasks:
2. LOOP instruction: Simple counter-based loops using CX
3. Conditional jumps: More flexible loop conditions
4. Nested loops: Require stack for counter preservation
5. Loop variants: LOOPE/LOOPNE for conditional looping
6. JCXZ: Efficient zero-check before entering loops

Understanding loops is crucial for array processing, string manipulation, and implementing algorithms in assembly language.

## Unconditional jump

In the 8086 microprocessor, an unconditional jump is an instruction that always transfers program control to a specified anywhere target address, regardless of any conditions or flags. When this instruction executes, the Instruction Pointer (IP) and sometimes the Code Segment (CS) register are modified to point to the new location, and the CPU continues fetching and executing instructions from that address.

## Syntax

**Jmp** Label\_Name/Pragraph\_Name

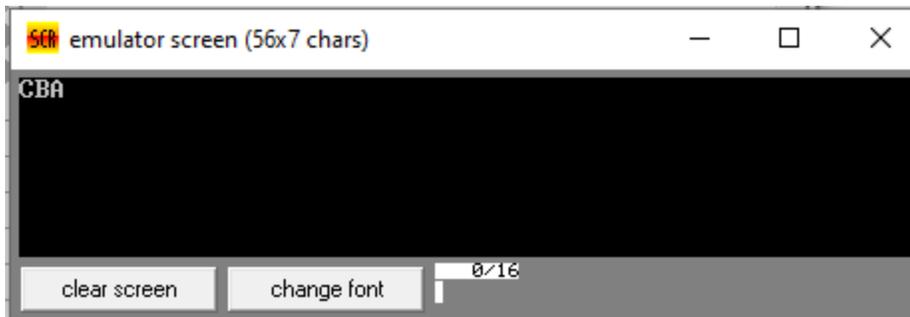
## Example

*; Unconditional jump to flow of program control from one point to another point*

```
.MODEL SMALL
.STACK 100H
.DATA
.CODE
MAIN PROC
    JMP DISPLAY_C: ; flow of control program jump from here to DISPLAY_C
DISPLAY_A:
    MOV AH,02H ; 02h DOS function to print A character
    MOV DX,'A'
    INT 21H ; Execute 02h function /service of Print one ASCII character

    INT 20H
DISPLAY_B:
    MOV AH,02H ; 02h DOS function to print A character
    MOV DX,'B'
    INT 21H ; Execute 02h function /service of Print one ASCII character
    JMP DISPLAY_A ; flow of control program jump from here to DISPLAY_A
DISPLAY_C:
    MOV AH,02H ; 02h DOS function to print A character
    MOV DX,'C'
    INT 21H ; Execute 02h function /service of Print one ASCII character
    JMP DISPLAY_B ; flow of control program jump from here to DISPLAY_B
MAIN ENDP

END MAIN
```



### Types of Unconditional Jumps in 8086

1. Short Jump (Relative Jump)
  - Range: -128 to +127 bytes from the current instruction
  - Opcode: EB (for 8-bit displacement)
2. Near Jump (Intra-segment Jump)
  - Range: Within the same code segment ( $\pm 32\text{KB}$  range)
  - Updates only: Instruction Pointer (IP)
  - Opcode: E9 (for 16-bit displacement)
3. Far Jump (Inter-segment Jump)
  - Range: To a different code segment
  - Updates both: CS and IP registers
  - Opcode: EA

### Summery or Key Points

1. No condition checking: The jump is taken 100% of the time
2. Different ranges: Short ( $\pm 128$  bytes), Near ( $\pm 32\text{KB}$ ), Far (anywhere in memory)
3. IP and CS changes: Near jumps change only IP, Far jumps change both CS and IP
4. Unconditional jumps are used for:
5. Creating loops (combined with conditional jumps)
6. Implementing switch/case structures
7. Jumping to subroutines (though CALL is usually preferred)
8. Error handling and interrupt service routines

The unconditional jump is fundamental for altering program flow and creating non-sequential execution paths in assembly language programs.

## Exercise

**Theory Questions.**

1. What is loop in computer programming perspectives?
2. How loop instruction work only explain.
3. Define unconditional jump.

**Practical Questions.**

1. Write assembly code to display odd and even series from 1 to 10 using loop instruction.

**Objective and MCQs:**

1. Which opcode of short jump.
  - a) EA
  - b) E9
  - c) EB
  - d) AB
2. Type of Near jumps change only \_\_\_\_\_ register.
  - a) DX
  - b) CX
  - c) IP
  - d) AX
3. The processor decreases the value of from \_\_\_\_\_ register automatically each time the loop runs.
  4. BX
  5. AX
  6. DX
  7. CX
4. Type of Far jumps change \_\_\_\_\_ and \_\_\_\_\_ register.
  5. AL , BL
  6. CS , BL
  7. CS, IP
  8. IP , DS
5. An unconditional jump is an instruction that always transfers program control to a specified target address.
  - a) Always top
  - b) Always bottom
  - c) Anywhere
  - d) Only top