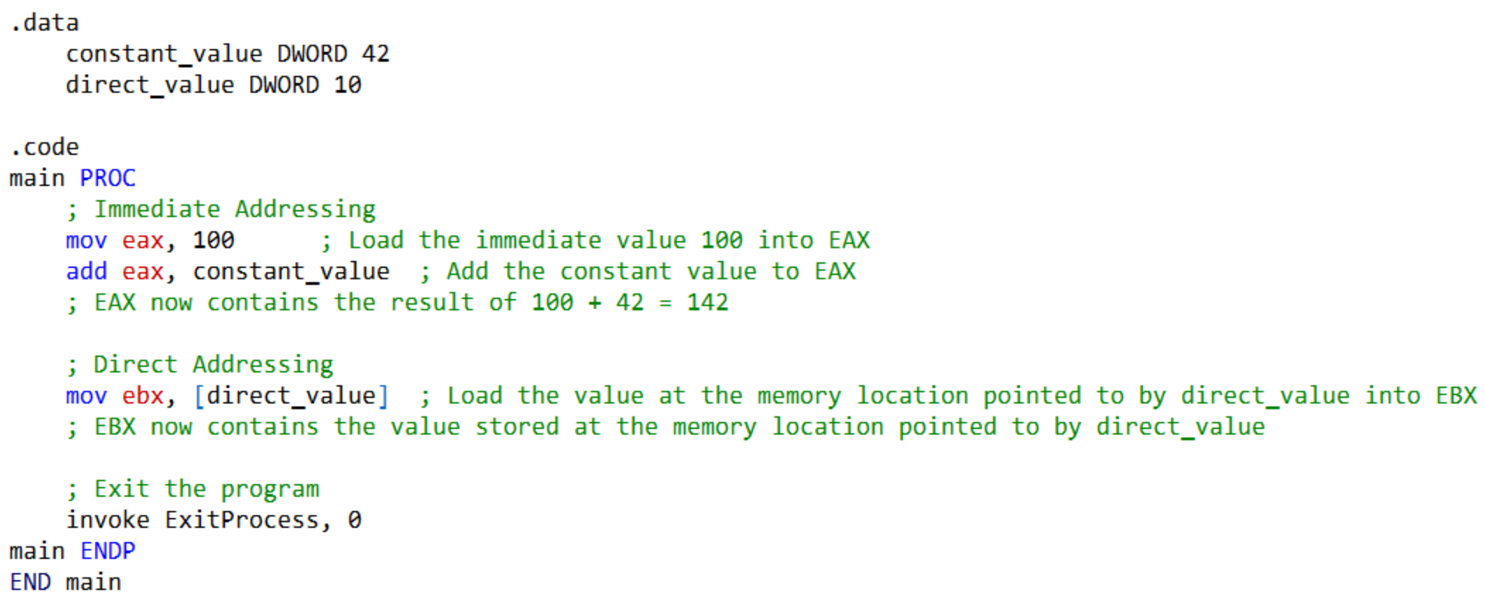
Immediate Addressing

Immediate addressing mode involves specifying a constant value as an operand directly within an instruction.

It's typically used for providing immediate data to instructions like add, subtract, or load.

In the immediate addressing mode, we specify the operand in the instruction itself.



```
.data
    constant_value DWORD 42
    direct_value DWORD 10

.code
main PROC
    ; Immediate Addressing
    mov eax, 100       ; Load the immediate value 100 into EAX
    add eax, constant_value  ; Add the constant value to EAX
    ; EAX now contains the result of 100 + 42 = 142

    ; Direct Addressing
    mov ebx, [direct_value]  ; Load the value at the memory location pointed to by direct_value into EBX
    ; EBX now contains the value stored at the memory location pointed to by direct_value

    ; Exit the program
    invoke ExitProcess, 0
main ENDP
END main
```

n this code:

Immediate addressing is demonstrated with mov eax, 100. Here, the immediate value 100 is directly loaded into the EAX register.

Direct addressing is demonstrated with mov ebx, [direct_value]. Here, the value at the memory location pointed to by the direct_value variable is loaded into the EBX register.

These two instructions showcase the difference between immediate addressing (where the value is directly specified in the instruction) and direct addressing (where the value is loaded from a memory location specified by a variable).

Differentiating the 5 addressing modes we've discussed:

```
.data
    array DWORD 10, 20, 30, 40  ; An array for indexed addressing
    value DWORD 100            ; A constant value for immediate addressing
    ptr DWORD offset array    ; A pointer for indirect addressing
    stackVal DWORD ?           ; A variable for stack addressing

.code
main PROC
    ; Direct Addressing
    mov eax, [array]   ; Load the value at the memory location pointed to by array into EAX

    ; Indirect Addressing
    mov ebx, [ptr]     ; Load the value at the memory location pointed to by ptr into EBX

    ; Indexed Addressing
    mov ecx, [array + 4]   ; Load the value at the memory location (array + 4) into ECX
    ; This demonstrates indexed addressing by accessing the second element of the array

    ; Immediate Addressing
    mov edx, value    ; Load the immediate value 100 into EDX

    ; Stack Addressing
    push 200          ; Push the immediate value 200 onto the stack
    pop stackVal      ; Pop the value from the stack into stackVal

    ; Exit the program
    invoke ExitProcess, 0
main ENDP
END main
```

In this code:

Direct addressing is demonstrated with mov eax, [array]. It loads the value at the memory location pointed to by array into EAX.

Indirect addressing is demonstrated with mov ebx, [ptr]. It loads the value at the memory location pointed to by ptr into EBX.

Indexed addressing is demonstrated with mov ecx, [array + 4]. It accesses the second element of the array by adding an offset of 4 bytes to array.

Immediate addressing is demonstrated with mov edx, value. It loads the immediate value 100 into EDX.

Stack addressing is demonstrated with push 200 and pop stackVal. It pushes the immediate value 200 onto the stack and then pops it into the stackVal variable.