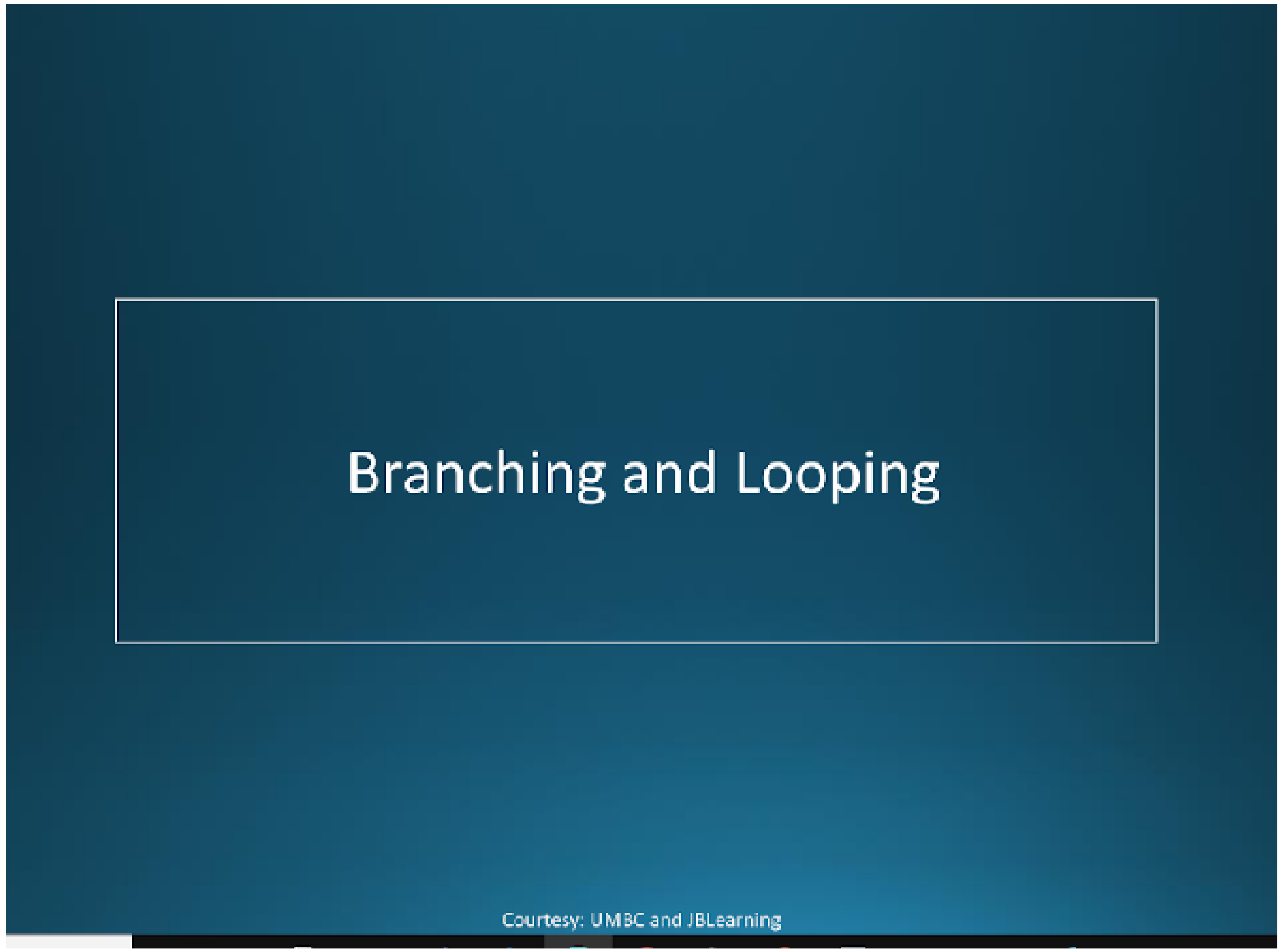
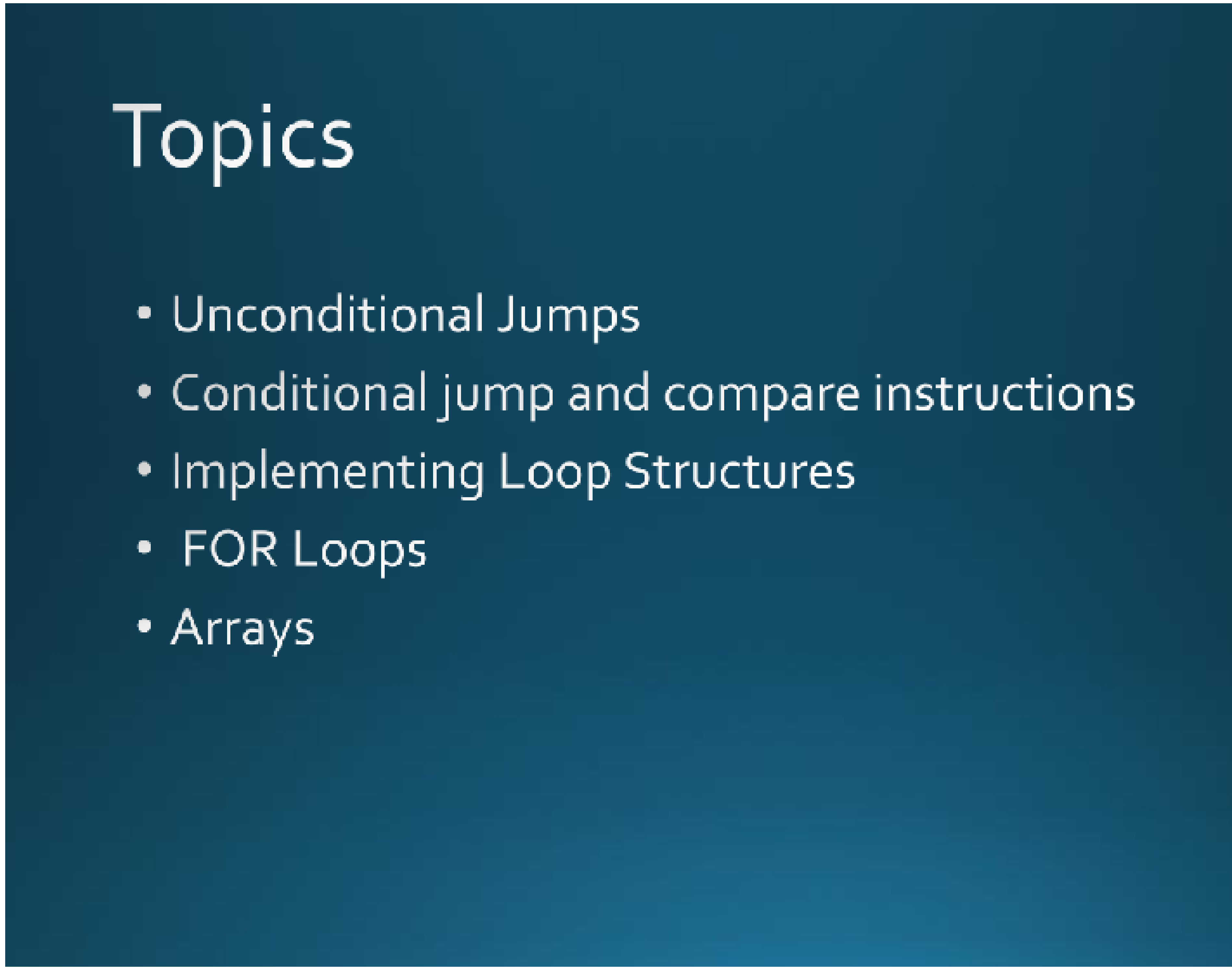


Branching and Looping

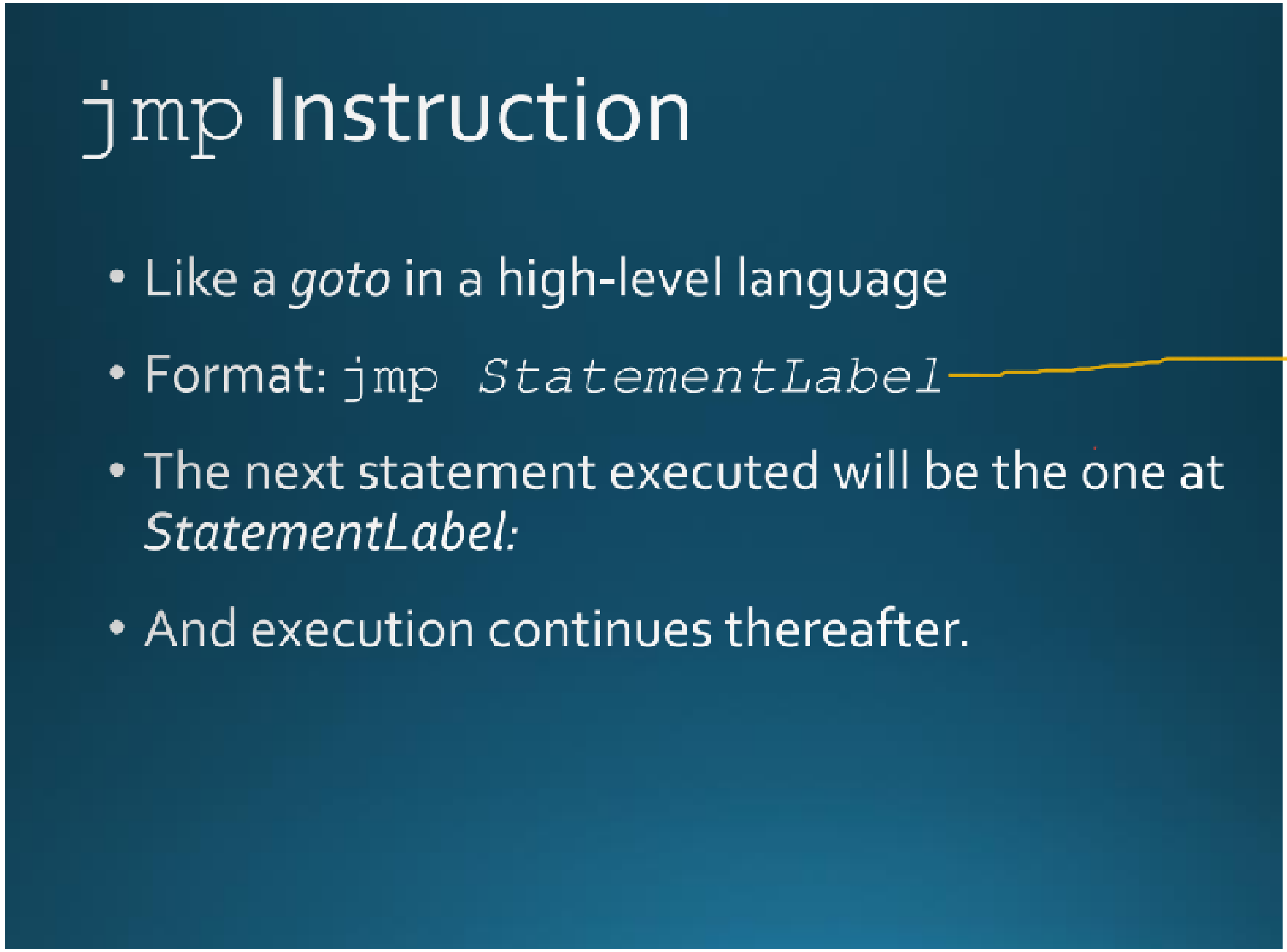
Courtesy: UMBC and JBLearning



Topics

- Unconditional Jumps
- Conditional jump and compare instructions
- Implementing Loop Structures
- FOR Loops
- Arrays

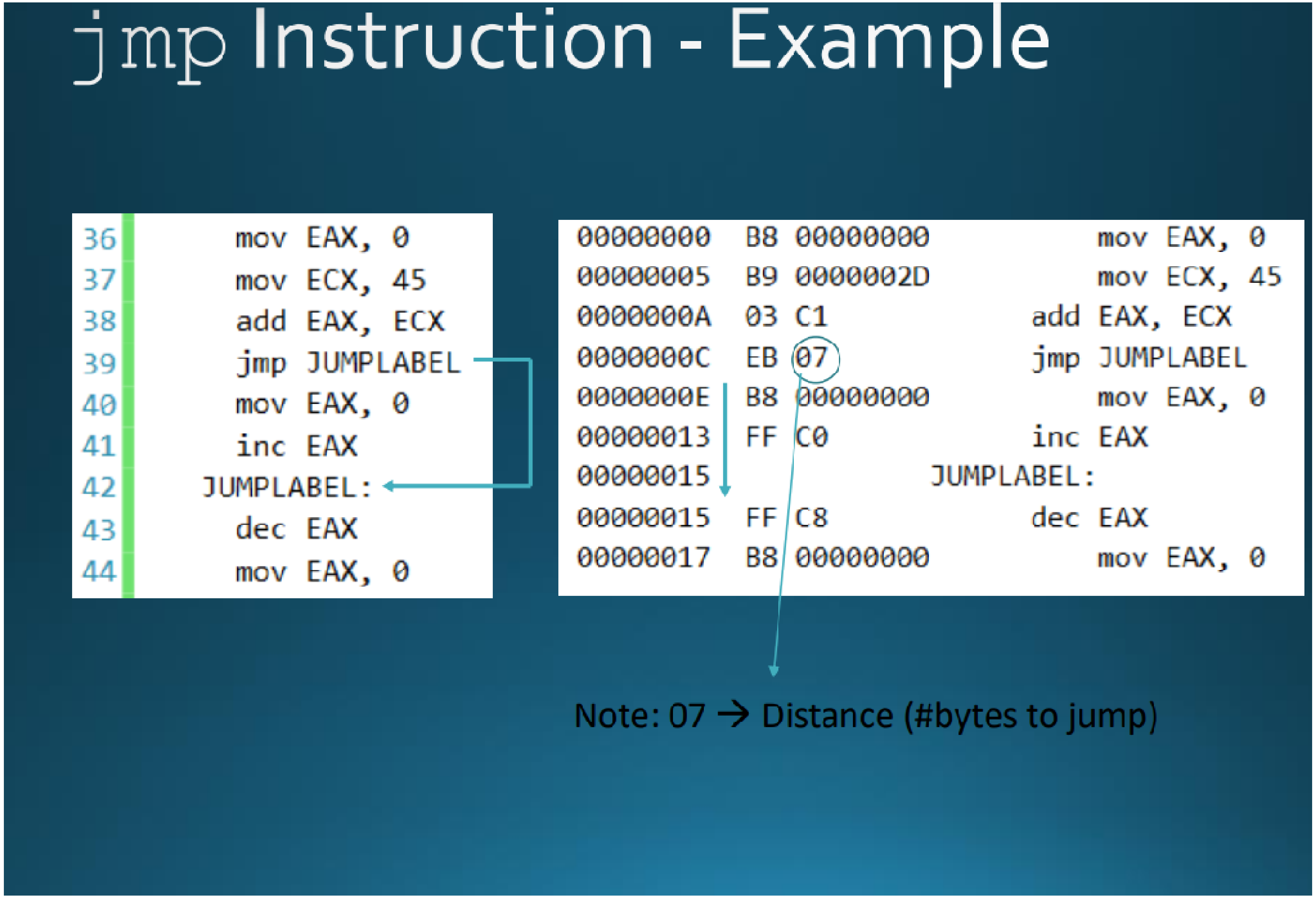
Unconditional Jumps



`jmp` Instruction

- Like a *goto* in a high-level language
- Format: `jmp` *`StatementLabel`*
- The next statement executed will be the one at *StatementLabel:*
- And execution continues thereafter.

Label



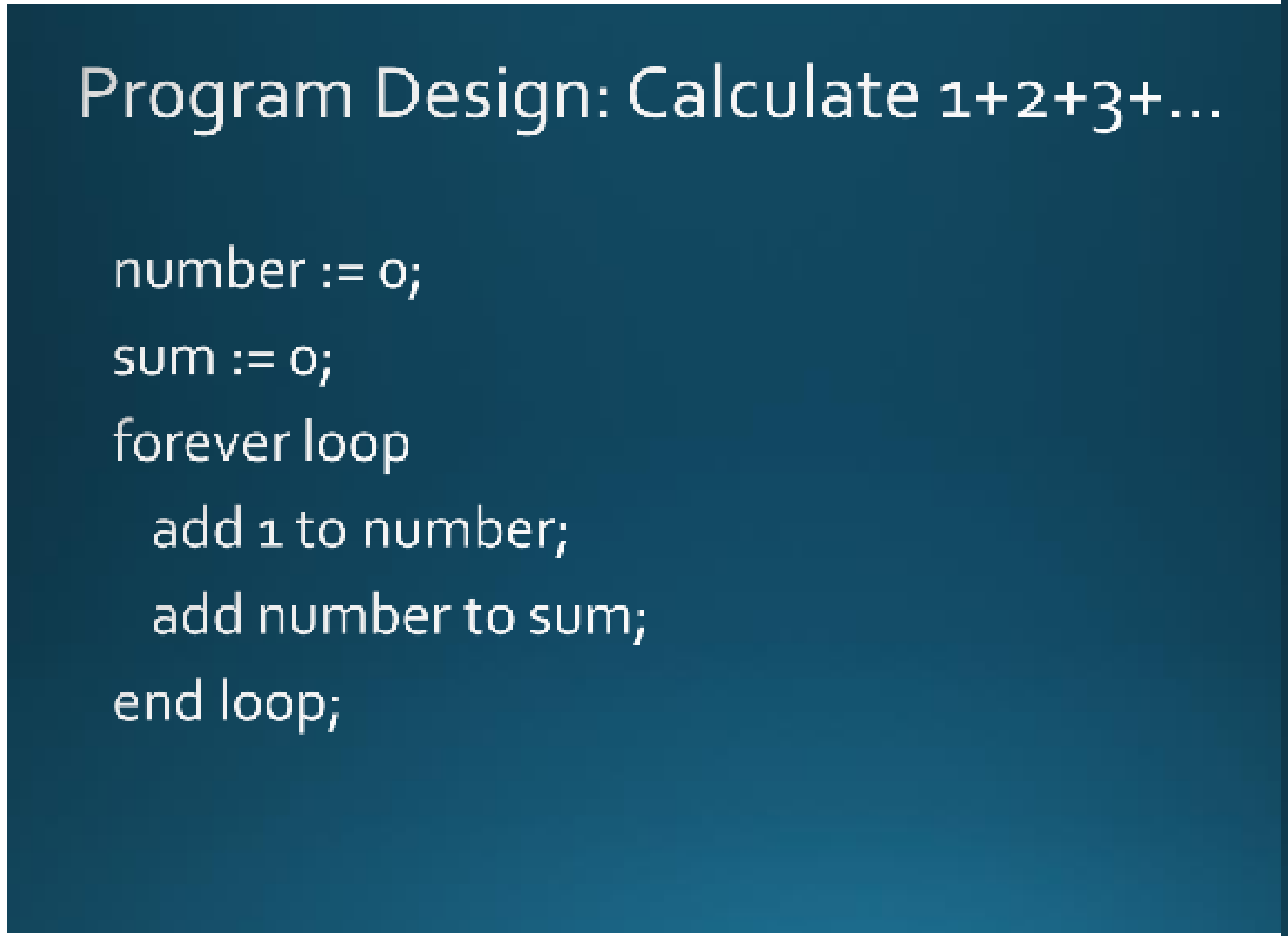
jmp Instruction - Example

```
        mov EAX, 0
        mov ECX, 45
        add EAX, ECX
        jmp JUMPLABEL
        mov EAX, 0
        inc EAX
    JUMPLABEL:
        dec EAX
        mov EAX, 0
```

```
00000000  B8 00000000          mov EAX, 0
00000005  B9 0000002D          mov ECX, 45
0000000A  03 C1            add EAX, ECX
0000000C  EB 07            jmp JUMPLABEL
0000000E  B8 00000000          mov EAX, 0
00000013  FF C0            inc EAX
00000015                 JUMPLABEL:
00000015  FF C8            dec EAX
00000017  B8 00000000          mov EAX, 0
```

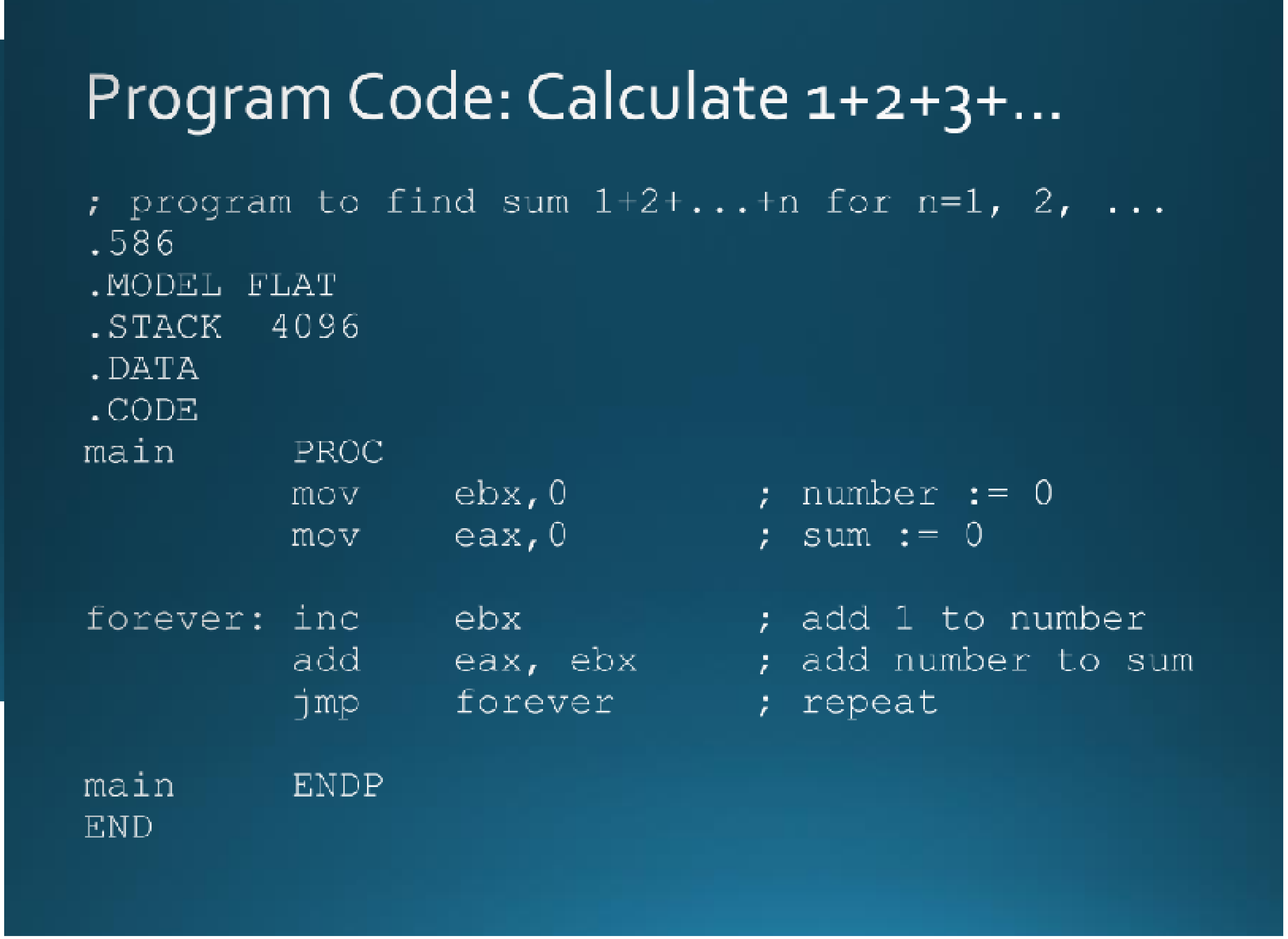
Note: 07 → Distance (#bytes to jump)

un conditional
jump



Program Design: Calculate 1+2+3+…

```
number := 0;
sum := 0;
forever loop
  add 1 to number;
  add number to sum;
end loop;
```



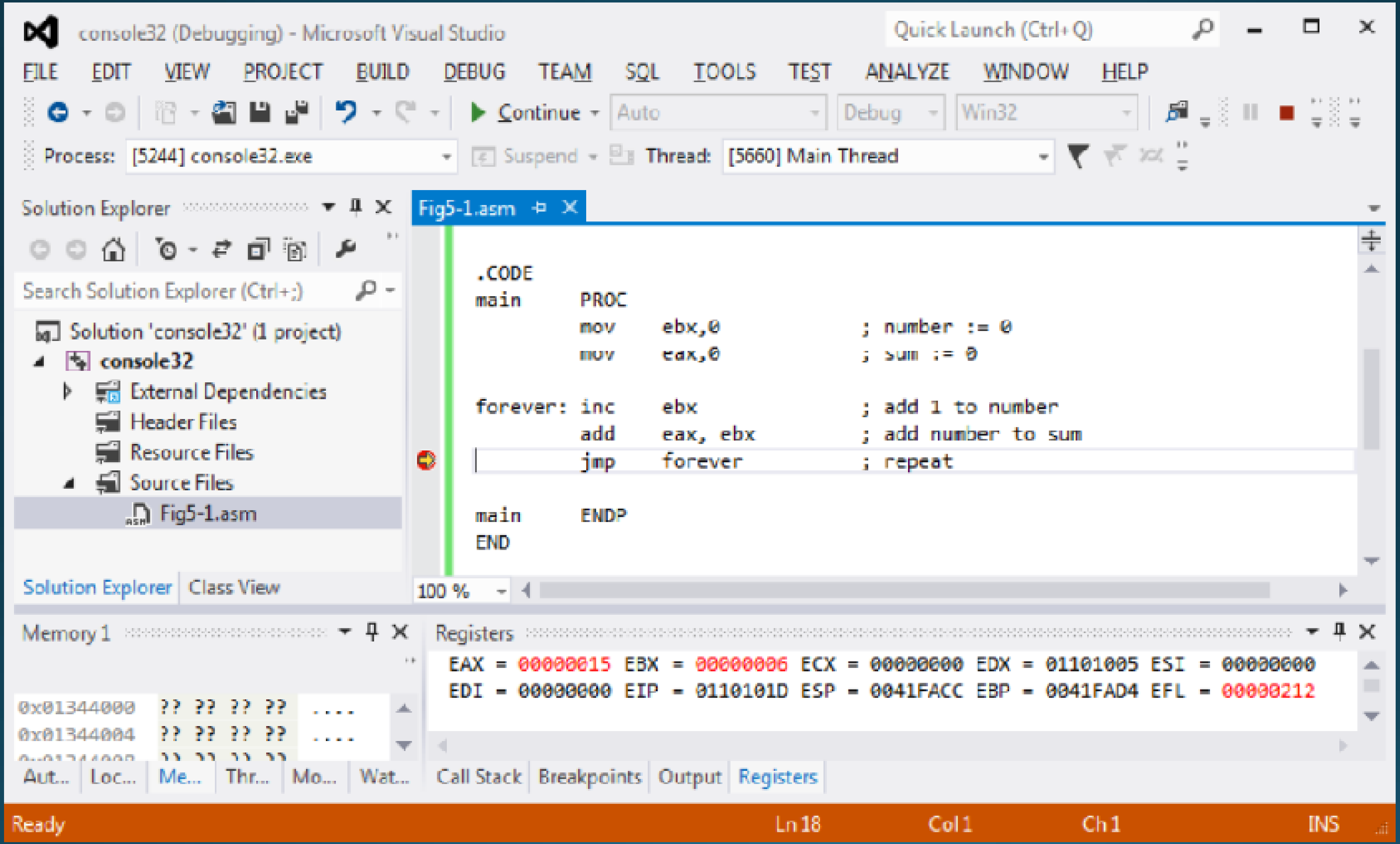
Program Code: Calculate 1+2+3+…

```
; program to find sum 1+2+...+n for n=1, 2, ...
.586
.MODEL FLAT
.STACK  4096
.DATA
.CODE
main      PROC
          mov     ebx,0          ; number := 0
          mov     eax,0          ; sum := 0

forever:  inc     ebx            ; add 1 to number
          add     eax, ebx       ; add number to sum
          jmp     forever        ; repeat

main      ENDP
END
```

Program Stopped at Breakpoint


console32 (Debugging) - Microsoft Visual Studio
Quick Launch (Ctrl+Q)
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL TOOLS TEST ANALYZE WINDOW HELP
Continue
Auto
Debug
Win32
Suspend
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'console32' (1 project)
console32
External Dependencies
Header Files
Resource Files
Source Files
Fig5-1.asm
Solution Explorer
Class View
100 %
Memory 1
Registers
Aut... Loc... Me... Thr... Mo... Wat...
Call Stack Breakpoints Output Registers
Ready
Ln 18
Col 1
Ch 1
INS

Process: [5244] console32.exe Thread: [5660] Main Thread

Fig5-1.asm

```
.CODE
main     PROC
         mov     ebx,0             ; number := 0
         mov     eax,0             ; sum := 0

forever: inc     ebx               ; add 1 to number
         add     eax, ebx          ; add number to sum
         jmp     forever           ; repeat

main     ENDP
END
```

```
EAX = 00000015 EBX = 00000006 ECX = 00000000 EDX = 01101005 ESI = 00000000
EDI = 00000000 EIP = 0110101D ESP = 0041FACC EBP = 0041FAD4 EFL = 00000212
```

```
0x01344000  ?? ?? ?? ??  ....
0x01344004  ?? ?? ?? ??  ....
```

jmp Reference - Direction

- Backward
  - To a statement that precedes the jmp statement
- Forward
  - To a statement that follows the jmp statement

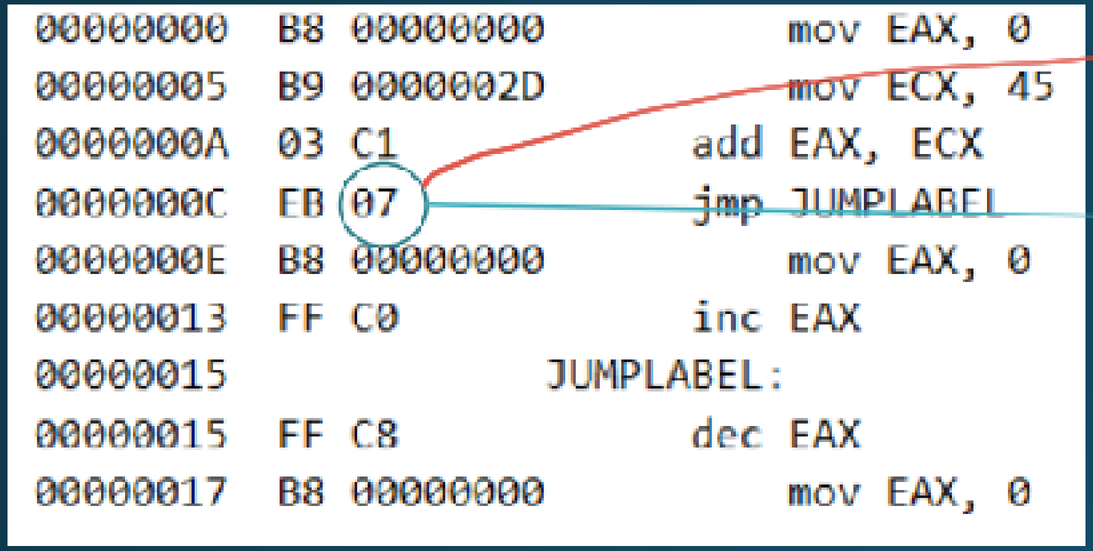
```
        mov EAX, 0
        mov ECX, 45
        add EAX, ECX
        jmp JUMPLABEL
        mov EAX, 0
        inc EAX
    JUMPLABEL:
        dec EAX
        mov EAX, 0
```

```
        mov     ebx,0           ; number := 0
        mov     eax,0           ; sum := 0

    forever:
        inc     ebx             ; add 1 to number
        add     eax, ebx        ; add number to sum
        jmp     forever         ; repeat
```

jmp Reference - Direction

- Forward

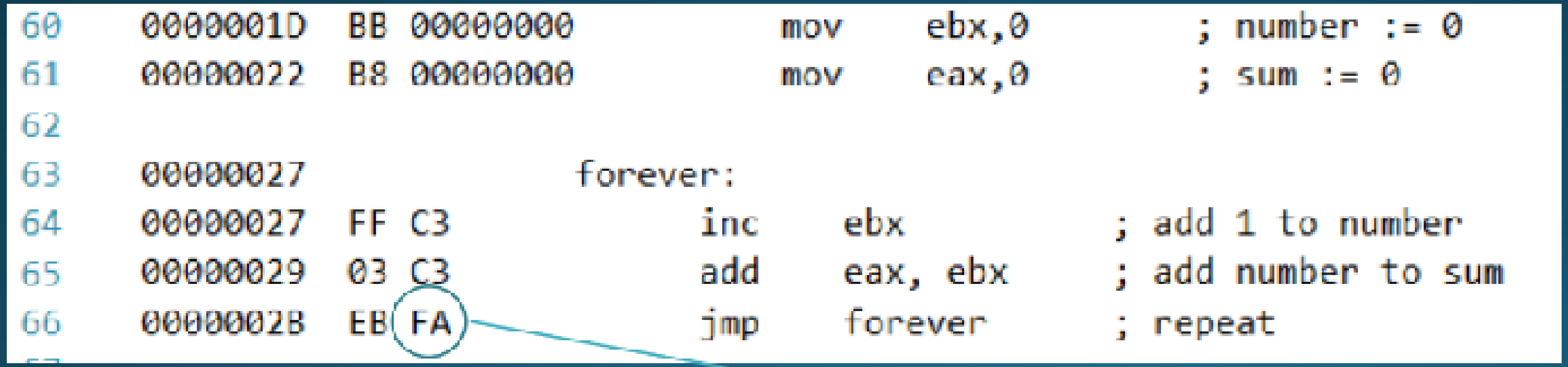


```
00000000  B8 00000000         mov EAX, 0
00000005  B9 0000002D         mov ECX, 45
0000000A  03 C1           add EAX, ECX
0000000C  EB 07           jmp JUMPLABEL
0000000E  B8 00000000         mov EAX, 0
00000013  FF C0           inc EAX
00000015              JUMPLABEL:
00000015  FF C8           dec EAX
00000017  B8 00000000         mov EAX, 0
```

Forward: 7 bytes(hex: 07)

-127 to 127
it is relativeshort

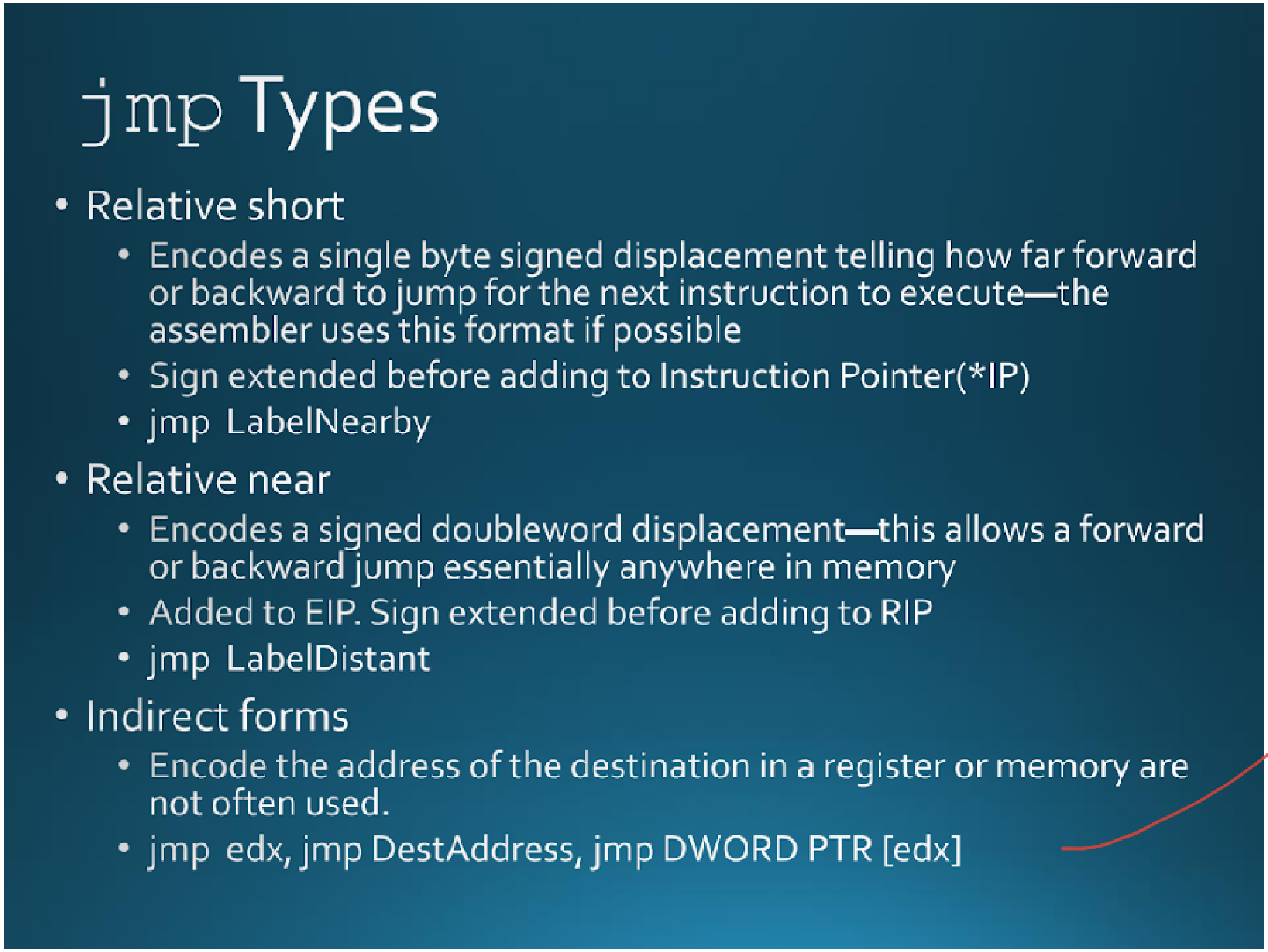
- Backward



```
0000001D  BB 00000000          mov     ebx,0          ; number := 0
00000022  B8 00000000          mov     eax,0          ; sum := 0

00000027              forever:
00000027  FF C3            inc     ebx          ; add 1 to number
00000029  03 C3            add     eax, ebx     ; add number to sum
0000002B  EB FA            jmp     forever      ; repeat
```

Backward: -6 bytes (hex: FA)



jmp Types

- Relative short
  - Encodes a single byte signed displacement telling how far forward or backward to jump for the next instruction to execute—the assembler uses this format if possible
  - Sign extended before adding to Instruction Pointer(*IP)
  - jmp LabelNearby
- Relative near
  - Encodes a signed doubleword displacement—this allows a forward or backward jump essentially anywhere in memory
  - Added to EIP. Sign extended before adding to RIP
  - jmp LabelDistant
- Indirect forms
  - Encode the address of the destination in a register or memory are not often used.
  - jmp edx, jmp DestAddress, jmp DWORD PTR [edx]

jmp ebx ---->register indirect

jmp varaddress
---->memory indirect

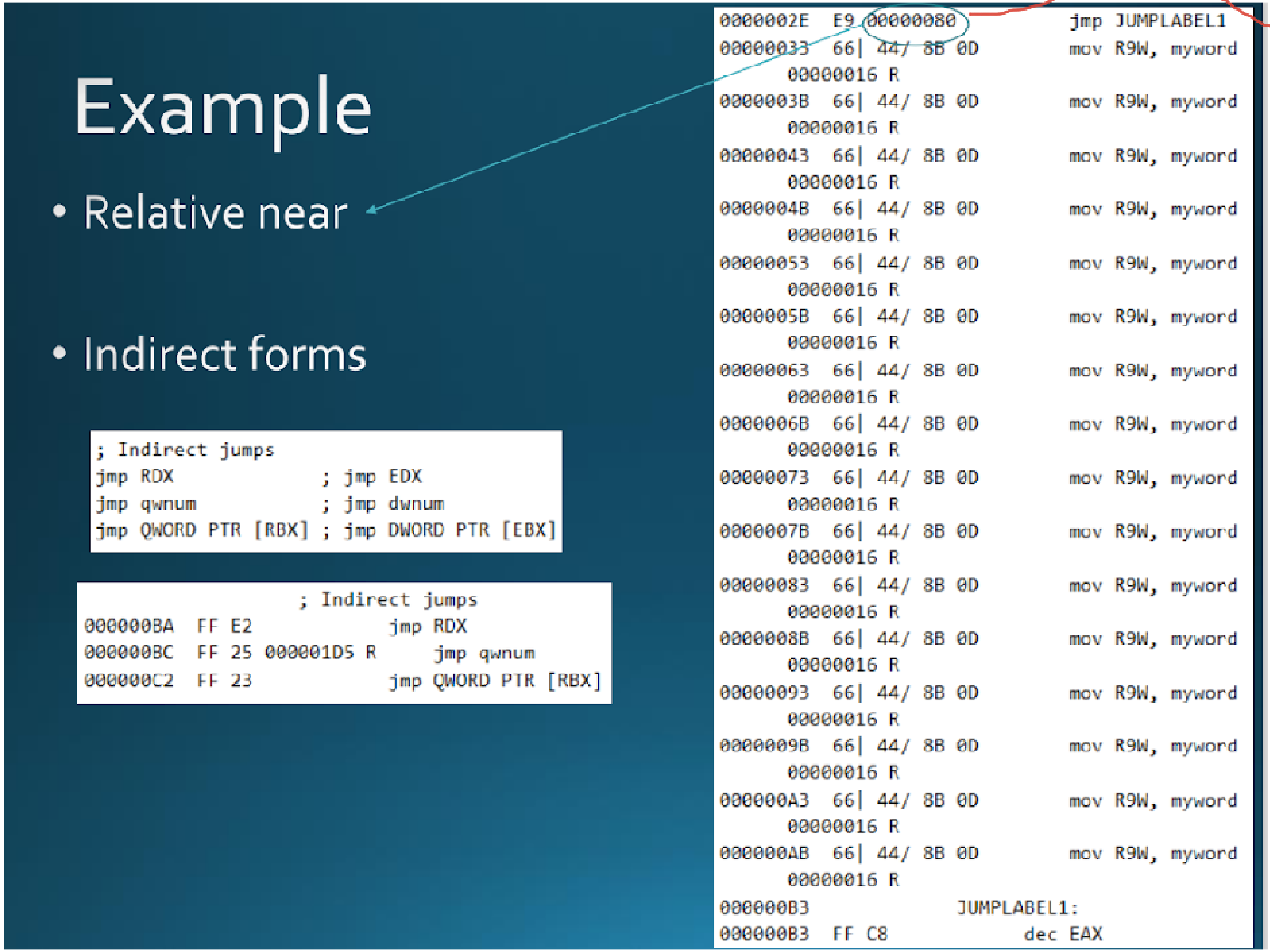
jmp dword ptr[ebx]
----------->memory indirect

relative short

relative near

memory indirect

register indirect



Example

- Relative near
- Indirect forms

```
; Indirect jumps
jmp RDX              ; jmp EDX
jmp qwnum            ; jmp dwnum
jmp QWORD PTR [RBX]  ; jmp DWORD PTR [EBX]
```

```
                  ; Indirect jumps
000000BA  FF E2             jmp RDX
000000BC  FF 25 000001D5 R      jmp qwnum
000000C2  FF 23             jmp QWORD PTR [RBX]
```

```
0000002E  E9 00000080           jmp JUMPLABEL1
00000033  66| 44/ 8B 0D         mov R9W, myword
       00000016 R
0000003B  66| 44/ 8B 0D         mov R9W, myword
       00000016 R
00000043  66| 44/ 8B 0D         mov R9W, myword
       00000016 R
0000004B  66| 44/ 8B 0D         mov R9W, myword
       00000016 R
00000053  66| 44/ 8B 0D         mov R9W, myword
       00000016 R
0000005B  66| 44/ 8B 0D         mov R9W, myword
       00000016 R
00000063  66| 44/ 8B 0D         mov R9W, myword
       00000016 R
0000006B  66| 44/ 8B 0D         mov R9W, myword
       00000016 R
00000073  66| 44/ 8B 0D         mov R9W, myword
       00000016 R
0000007B  66| 44/ 8B 0D         mov R9W, myword
       00000016 R
00000083  66| 44/ 8B 0D         mov R9W, myword
       00000016 R
0000008B  66| 44/ 8B 0D         mov R9W, myword
       00000016 R
00000093  66| 44/ 8B 0D         mov R9W, myword
       00000016 R
0000009B  66| 44/ 8B 0D         mov R9W, myword
       00000016 R
000000A3  66| 44/ 8B 0D         mov R9W, myword
       00000016 R
000000AB  66| 44/ 8B 0D         mov R9W, myword
       00000016 R
000000B3             JUMPLABEL1:
000000B3  FF C8            dec EAX
```

80--->128 in decimal

-127 to 127----> relative short

else relative near

jmp Encoding

Type	Opcode	#bytes of obj code
relative near	E9	5
relative short	EB	2
register indirect	FF	2
memory indirect	FF	2+