

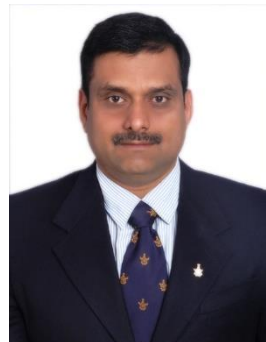


# **Microprocessors and Interfaces: 2021-22**

## **Lecture 16**

### **8086 Branching & Program Control Instructions : Part-1**

**By Dr. Sanjay Vidhyadharan**



# THE JUMP GROUP

- Allows programmer to skip program sections and branch to any part of memory for the next instruction.
- A conditional jump instruction allows decisions based upon numerical tests.
  - results are held in the flag bits, then tested by conditional jump instructions
- LOOP is also a form of the jump instruction.

# Branching Instructions

- **Conditional Jump**

- JC/JNC            —→ Carry
- JZ/JNZ           —→ Zero
- JP/JNP           —→ Parity
- JS/JNS           —→ Sign
- JO/JNO           —→ Overflow
- JCXZ             —→ CX = 0
- JE/JNE

# Branching Instruction : Conditional Jump

Mnemonic	Meaning	Jump Condition
JA	Jump if Above	CF=0 and ZF=0
JAE	Jump if Above or Equal	CF=0
JB	Jump if Below	CF=1
JBE	Jump if Below or Equal	CF=1 or ZF=1
JC	Jump if Carry	CF=1
JCXZ	Jump if CX Zero	CX=0
JE	Jump if Equal	ZF=1
JG	Jump if Greater (signed)	ZF=0 and SF=OF
JGE	Jump if Greater or Equal (signed)	SF=OF
JL	Jump if Less (signed)	SF != OF
JLE	Jump if Less or Equal (signed)	ZF=1 or SF != OF
JMP	Unconditional Jump	unconditional
JNA	Jump if Not Above	CF=1 or ZF=1
JNAE	Jump if Not Above or Equal	CF=1
JNB	Jump if Not Below	CF=0
JNBE	Jump if Not Below or Equal	CF=0 and ZF=0

# Branching Instruction : Conditional Jump

Mnemonic	Meaning	Jump Condition
JNC	Jump if Not Carry	CF=0
JNE	Jump if Not Equal	ZF=0
JNG	Jump if Not Greater (signed)	ZF=1 or SF != OF
JNGE	Jump if Not Greater or Equal (signed)	SF != OF
JNL	Jump if Not Less (signed)	SF=OF
JNLE	Jump if Not Less or Equal (signed)	ZF=0 and SF=OF
JNO	Jump if Not Overflow (signed)	OF=0
JNP	Jump if No Parity	PF=0
JNS	Jump if Not Signed (signed)	SF=0
JNZ	Jump if Not Zero	ZF=0
JO	Jump if Overflow (signed)	OF=1
JP	Jump if Parity	PF=1
JPE	Jump if Parity Even	PF=1
JPO	Jump if Parity Odd	PF=0
JS	Jump if Signed (signed)	SF=1
JZ	Jump if Zero	ZF=1

# Branching Instruction : Conditional Jump

- **Conditional Jump**

**Unsigned numbers:**

JA

JAE

JB

JBE

**Signed numbers:**

JG

JGE

JL

JLE

# Branching Instruction : Conditional Jump

## Example 1

CMP AX, 0030H; compares by subtracting 0030H from the value in AX register  
JA LABEL1; jumps to the address specified by LABEL1 if value in register AX is above the value 0030H

## Example 2

CMP AX, 0030H; compares by subtracting 0030H from the value in AX register  
JAE LABEL1; jumps to the address specified by LABEL1 if value in register AX is above or equal to the value 0030H

## Example 3

CMP AX, 0030H; compares by subtracting 0030H from the value in AX register  
JB LABEL1; jumps to the address specified by LABEL1 if value in register AX is below the value 0030H

# Branching Instruction : Conditional Jump

All conditional jumps have one big limitation, unlike **JMP** instruction they can only jump **127** bytes forward and **128** bytes backward (note that most instructions are assembled into 3 or more bytes).

<b>JE/JZ</b> = Jump on Equal/Zero	01110100	disp
<b>JL/JNGE</b> = Jump on Less/Not Greater or Equal	01111100	disp
<b>JLE/JNG</b> = Jump on Less or Equal/Not Greater	01111110	disp
<b>JB/JNAE</b> = Jump on Below/Not Above or Equal	01110010	disp
<b>JBE/JNA</b> = Jump on Below or Equal/Not Above	01110110	disp
<b>JP/JPE</b> = Jump on Parity/Parity Even	01111010	disp
<b>JO</b> = Jump on Overflow	01110000	disp
<b>JS</b> = Jump on Sign	01111000	disp



# Branching Instruction : Conditional Jump

## Example 1:

```

Mov AX, 0030H;
CMP AX, 0030H;
JE Label1;
Mov BX, 0000H;
Label1: Mov BX, 0001H;
    
```

registers	H	L
AX	00	30
BX	00	00
CX	00	0F
DX	00	00
CS	0700	
IP	0106	
SS	0700	
SP	FFFE	

0700:0106		0700:0106	
07100:	B8 184	MOV AX, 00030h	
07101:	30 048 0	CMP AX, 00030h	
07102:	00 000 NULL	JZ 010Bh	
07103:	3D 061 =	MOV BX, 00000h	
07104:	30 048 0	MOV BX, 00001h	
07105:	00 000 NULL	RET	
07106:	74 116 t	NOP	
07107:	03 003 v	NOP	
07108:	BB 187	NOP	
07109:	00 000 NULL	NOP	
0710A:	00 000 NULL	NOP	
0710B:	BB 187	NOP	
0710C:	01 001	NOP	
0710D:	00 000 NULL	NOP	

**JE/JZ = Jump on Equal/Zero**  
**JL/JNGE = Jump on Less/Not**

01110100      disp  
 011111100      disp

# Branching Instruction : Conditional Jump

## Example 2:

Mov AX, 0030H;

Label1:

Mov BX, 0000H;

CMP AX, BX;

JE Label1;

Mov BX, 0001H;

The screenshot shows an emulator window titled 'emulator: noname.com\_'. The menu bar includes 'file', 'math', 'debug', 'view', 'external', 'virtual devices', 'virtual drive', and 'help'. Below the menu are buttons for 'Load', 'reload', 'step back', 'single step', 'run', and a 'step delay ms: 0' slider. The 'run' button is highlighted. On the left, a 'registers' panel shows the state of various registers: AX (00 30), BX (00 00), CX (00 0E), DX (00 00), CS (07 00), IP (01 08), SS (07 00), SP (FFFE), and BP (0000). The main window displays assembly code with memory addresses and values. The instruction 'JZ 0103h' is highlighted in blue. The code includes: MOV AX, 00030h; MOV BX, 00000h; CMP AX, BX; JZ 0103h; MOV BX, 00001h; RET; followed by several NOP instructions.

JE/JZ = Jump on Equal/Zero

01110100

disp

JL/JNGE = Jump on Less/Not

01111100

disp

# Branching Instruction : Unconditional Jump

## Unconditional jump Instructions

- Short or Near jump or **Intra segment** jump
- Far or **Intersegment** jump
- Near and Far jumps are further divided into **Direct** or **Indirect**
  - Direct -Destination address specified as a part of the instruction
  - Indirect-Destination address specified in a register or memory location

Sanjay Vidhyadharan

# Branching Instruction : Unconditional Jump

## Short Jump

- If the target label (address) is within  $-128$  to  $127$  locations (bytes) of the instruction following the JMP (remember, the offset is added to the current value of the IP, which is pointing to the next instruction)
- It is assembled as a SHORT instruction (2 bytes). Only eight bits are needed to specify the address (these eight bits are added to the IP)

### JMP = Unconditional Jump:

Direct Within Segment	11101001	disp-low	disp-high
Direct Within Segment-short	11101011	disp	
Indirect Within Segment	11111111	mod 100 r/m	
Direct Intersegment	11101010	offset-low sea-low	offset-high sea-high

# Branching Instruction : Unconditional Jump

## Short Jump

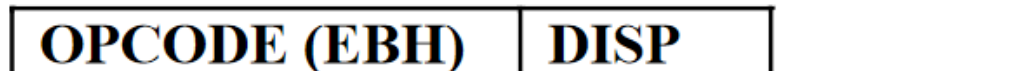
### Example

Offset	Machine Code	Source Code
0100	B4 02	start: mov ah, 2 ;loop start
0102	B2 41	mov dl, 'A' ;
0104	CD 21	int 21h ;disp A
0106	EB F8	jmp start ;jmp back
0108	.....	(rest of program)

How does the compiler know it's a SHORT jump?

$$0100 - 0108 = -8 = F8$$

**Short JMP**



# Branching Instruction : Unconditional Jump

## Near Jump

3-byte **Near jump** allows a branch or jump within  $\pm 32\text{K}$  bytes from the instruction in the current code segment.

### **JMP = Unconditional Jump:**

Direct Within Segment	11101001	disp-low	disp-high
Direct Within Segment-short	11101011	disp	
Indirect Within Segment	11111111	mod 100 r/m	
Direct Intersegment	11101010	offset-low seg-low	offset-high seg-high

Short and Near jumps are relocatable because they are relative jump.

# Branching Instruction : Unconditional Jump

Example: Encodings of short, near, and far jumps.

0005	33 C0		XOR AX, AX
0007	40	Back:	INC AX
0008	EB 10		JMP Forward
000A	B9 000A		MOV CX, 10
000D	E9 000A		JMP Near PTR Forward
0010	B9 0014		MOV cx, 20
0013	EA ---- 001A R		JMP Far PTR Forward
0018	8B C1		MOV AX, CX
001A	03 C0	Forward:	ADD AX, AX
001C	EB E9		JMP Back

**Short JMP**

**Near JMP**

**Intersegment JMP**

<b>OPCODE (EBH)</b>	<b>DISP</b>			
<b>OPCODE (E9H)</b>	<b>IP Low</b>	<b>IP High</b>		
<b>OPCODE (EAH)</b>	<b>IP Low</b>	<b>IP High</b>	<b>CS Low</b>	<b>CS High</b>

# Branching Instruction : Unconditional Jump

**Indirect Program Memory Addressing** • If a 16-bit register holds the address of a JMP instruction, the jump is near.

For example, if the BX register contains 1000H and a JMP BX instruction executes, the microprocessor jumps to offset address 1000H in the current code segment.

## **JMP = Unconditional Jump:**

Direct Within Segment	11101001	disp-low	disp-high
Direct Within Segment-short	11101011	disp	
Indirect Within Segment	11111111	mod 100 r/m	
Direct Intersegment	11101010	offset-low seg-low	offset-high seg-high
Indirect Intersegment	11111111	mod 101 r/m	



# Branching Instruction : Unconditional Jump

## Far Jump

- 5-byte **far jump** allows a jump to any memory location within the real memory system.
- The short and near jumps are often called **intra-segment jumps**.
- Far jumps are called **inter-segment jumps**.

Indirect Intersegment	11111111	mod 011 r/m	
<b>JMP = Unconditional Jump:</b>			
Direct Within Segment	11101001	disp-low	disp-high
Direct Within Segment-short	11101011	disp	
Indirect Within Segment	11111111	mod 100 r/m	
Direct Intersegment	11101010	offset-low	offset-high
		seg-low	seg-high
Indirect Intersegment	11111111	mod 101 r/m	

# Branching Instruction : Unconditional Jump

Example: Encodings of short, near, and far jumps.

0005	33	C0		XOR AX, AX
0007	40		Back:	INC AX
0008	EB	10		JMP Forward
000A	B9	000A		MOV CX, 10
000D	E9	000A		JMP Near PTR Forward
0010	B9	0014		MOV cx, 20
0013	EA	----	001A R	JMP Far PTR Forward
0018	8B	C1		MOV AX, CX
001A	03	C0	Forward:	ADD AX, AX
001C	EB	E9		JMP Back

Short JMP

Near JMP

Intersegment JMP

<b>OPCODE (EBH)</b>	<b>DISP</b>			
<b>OPCODE (E9H)</b>	<b>IP Low</b>	<b>IP High</b>		
<b>OPCODE (EAH)</b>	<b>IP Low</b>	<b>IP High</b>	<b>CS Low</b>	<b>CS High</b>

**Thankyou**