Computer Organization & Assembly Language
(Lecture 08)

Omar Bin Samin
Lecturer
Institute of Management Sciences, Peshawar.

Jump Instructions

Jump Instructions

- Jump Instructions are used for changing the flow of execution of instructions in the processor

- There are two types of Jump instructions:
  1. Unconditional Jump Instructions
  2. Conditional Jump Instructions

Unconditional Jump Instructions

- These instructions are used to jump on a particular location unconditionally, i.e. there is no need to satisfy any condition for the jump to take place

- Syntax:

jmp tag

Example

- Show the values of AX and BX for all executable instructions.

```
[org 0x0100]
mov ax,0x4D2
mov bx,0x38
add bh,al
jmp tag1
jmp tag2
sub ah,0xFF
tag1:
mov ax,0
tag2:
add ax,0xFFFF
mov ax,0x4c00
int 0x21
```

Solution

Syntax	AX		BX	
	AH	AL	BH	BL
[org 0x100]				
Mov ax, 1234	04	D2	00	00
Mov bx, 56	04	D2	00	38
Add bh,al	04	D2	D2	38
Jmp Tag1	04	D2	D2	38
Jmp Tag2				
Sub ah, 0FFH				
Tag1:				
Mov ax, 0	00	00	D2	38
Tag2:				
Add ax, 0FFFFH	FF	FF	D2	38
Ret				

Task 01

- Show the values of AX and BX for all executable instructions.

```
[org 0x0100]
mov ax,0x4D2
mov bx,0x38
add bh,al
jmp tag1
jmp tag2
sub ah,0xFF
tag1:
mov ax,0
mov ax,0x4c00
int 0x21
tag2:
add ax,0xFFFF
mov ax,0x4c00
int 0x21
```

Task 02

- Show the values of AX and BX for all executable instructions.

```
[org 0x0100]
mov ax,1
mov bx,5
jmp tag3
add bh,al
jmp tag1
jmp tag2
sub ah,0xFF
tag1:
mov ax,0
tag3:
tag2:
add ax,0xFFFF
mov ax,0x4c00
int 0x21
```

Conditional Jump Instruction

- In these types of instructions, the processor must check for the particular condition. If it is true, only then jump takes place, else the normal flow of program execution is maintained
- The arithmetic and logic operations set flags
- The conditional jump statements test the flags and jump if the relevant flag is set

- Conditional jump instructions are as follows:
  - JE / JZ
  - JNE / JNZ
  - JC / JB
  - JNC / JNB
  - JS
  - JNS
  - JO
  - JNO
  - JP / JPE
  - JNP / JPO

JE / JZ

- JE stands for ‘Jump if Equal’
- JZ stands for ‘Jump if Zero’
- It checks whether the Zero Flag is set or not
  - If ZF = 1, then jump
  - Else, do nothing

JNE / JNZ

- JNE stands for ‘Jump if Not Equal’
- JNZ stands for ‘Jump if Not Zero’
- It checks whether the Zero Flag is set or not
  - If ZF = 0, then jump
  - Else, do nothing

JC / JB

- JC stands for ‘Jump if Carry’
- JB stands for ‘Jump if Below’
- It checks whether the Carry Flag is set or not
  - If CF = 1, then jump
  - Else, do nothing

JNC / JNB

- JNC stands for ‘Jump if Not Carry’
- JNB stands for ‘Jump if Not Below’
- It checks whether the Carry Flag is set or not
  - If CF = 0, then jump
  - Else, do nothing

JS

- JS stands for ‘Jump if Sign’
- It checks whether the Sign Flag is set or not
  - If SF = 1, then jump
  - Else, do nothing

JNS

- JNS stands for ‘Jump if Not Sign’
- It checks whether the Sign Flag is set or not
  - If SF = 0, then jump
  - Else, do nothing

JO

- JO stands for ‘Jump if Overflow’
- It checks whether the Overflow Flag is set or not
  - If OF = 1, then jump
  - Else, do nothing

JNO

- JNO stands for ‘Jump if Not Overflow’
- It checks whether the Overflow Flag is set or not
  - If OF = 0, then jump
  - Else, do nothing

JP / JPE

- JP stands for ‘Jump if Parity’
- JPE stands for ‘Jump if Even Parity’
- It checks whether the Parity Flag is set or not
  - If PF = 1, then jump
  - Else, do nothing

JNP / JPO

- JNP stands for ‘Jump if Not Parity’
- JPO stands for ‘Jump if Odd Parity’
- It checks whether the Parity Flag is set or not
  - If PF = 0, then jump
  - Else, do nothing

- Conditional jump instructions for **Signed Numbers** are as follows:
  - JG / JNLE
  - JNG / JLE
  - JL / JNGE
  - JNL / JGE

JG / JNLE

- JG stands for ‘Jump if Greater’
- JNLE stands for ‘Jump if Not Less or Equal’
  - If ZF = 0 and SF = OF, then jump
  - Else, do nothing

JNG / JLE

- JNG stands for ‘Jump if Not Greater’
- JLE stands for ‘Jump if Less or Equal’
  - If ZF = 1 and SF != OF, then jump
  - Else, do nothing

JL / JNGE

- JL stands for ‘Jump if Less’
- JNGE stands for ‘Jump if Not Greater or Equal’
  - If SF != OF, then jump
  - Else, do nothing

JNL / JGE

- JNL stands for ‘Jump if Not Less’
- JGE stands for ‘Jump if Greater or Equal’
  - If SF = OF, then jump
  - Else, do nothing

- Conditional jump instructions for **Un-signed Numbers** are as follows:
  - JA / JNBE
  - JNA / JBE
  - JB / JNAE
  - JNB / JAE

JA / JNBE

- JA stands for ‘Jump if Above’
- JNBE stands for ‘Jump if Not Below or Equal’
  - If CF = 0 and ZF = 0, then jump
  - Else, do nothing

JNA / JBE

- JNA stands for ‘Jump if Not Above’
- JBE stands for ‘Jump if Below or Equal’
  - If CF = 1 and ZF = 1, then jump
  - Else, do nothing

JB / JNAE

- JB stands for ‘Jump if Below’
- JNAE stands for ‘Jump if Not Above or Equal’
  - If CF = 1, then jump
  - Else, do nothing

JNB / JAE

- JNB stands for ‘Jump if Not Below’
- JAE stands for ‘Jump if Above or Equal’
  - If CF = 0, then jump
  - Else, do nothing

Example

- Show the values of AX, BX, ZF, OF, SF, AF, PF and CF for all executable instructions.

```
[org 0x0100]
mov al,2
mov bl,2
cmp al,bl
jne Tag2
je Tag1
jmp Tag2
Tag1:
add al, 0FFH
jnc Tag2
jc Tag3
mov ax,0x4c00
int 0x21
Tag2:
sub al,bl
mov ax,0x4c00
int 0x21
Tag3:
mov ax, 0
mov ax,0x4c00
int 0x21
```

Solution

Syntax	AX		BX		ZF	OF	SF	AF	PF	CF	IF
	AH	AL	BH	BL							
Org 100h											
Mov al, 2	00	02	00	00	0	0	0	0	0	0	1
Mov bl, 2	00	02	00	02	0	0	0	0	0	0	1
cmp al, bl	00	02	00	02	1	0	0	0	1	0	1
Jne Tag2	00	02	00	02	1	0	0	0	1	0	1
Je Tag1	00	02	00	02	1	0	0	0	1	0	1
Jmp Tag2											
Tag1:											
Add al, 0FFH	00	01	00	02	0	0	0	1	0	1	1

Syntax	AX		BX		ZF	OF	SF	AF	PF	CF	IF
	AH	AL	BH	BL							
Jnc Tag2	00	01	00	02	0	0	0	1	0	1	1
Jc Tag3	00	01	00	02	0	0	0	1	0	1	1
Ret											
Tag2:											
Sub al,bl											
Ret											
Tag3:											
Mov ax, 0	00	00	00	02	0	0	0	1	0	1	1
Ret											

Task 03

- Show the values of AX, BX, ZF, OF, SF, AF, PF and CF for all executable instructions.

```
[org 0x0100]
mov al,0x80
mov bl,0x4
add al,bl
jg Tag1
ja Tag2
jl Tag3
jb Tag4
Tag1:
mov ax,0x1111
mov ax,0x4c00
int 0x21
Tag2:
mov ax,0x2222
mov ax,0x4c00
int 0x21
Tag3:
mov ax,0x3333
mov ax,0x4c00
int 0x21
Tag4:
mov ax,0x4444
mov ax,0x4c00
int 0x21
```

Task 04

- Show the values of AX, BX, ZF, OF, SF, AF, PF and CF for all executable instructions.

```
[org 0x0100]
mov al,0x80
mov bl,0x4
sub al,bl
jg Tag1
ja Tag2
jl Tag3
jb Tag4
Tag1:
mov ax,0x1111
mov ax,0x4c00
int 0x21
Tag2:
mov ax,0x2222
mov ax,0x4c00
int 0x21
Tag3:
mov ax,0x3333
mov ax,0x4c00
int 0x21
Tag4:
mov ax,0x4444
mov ax,0x4c00
int 0x21
```