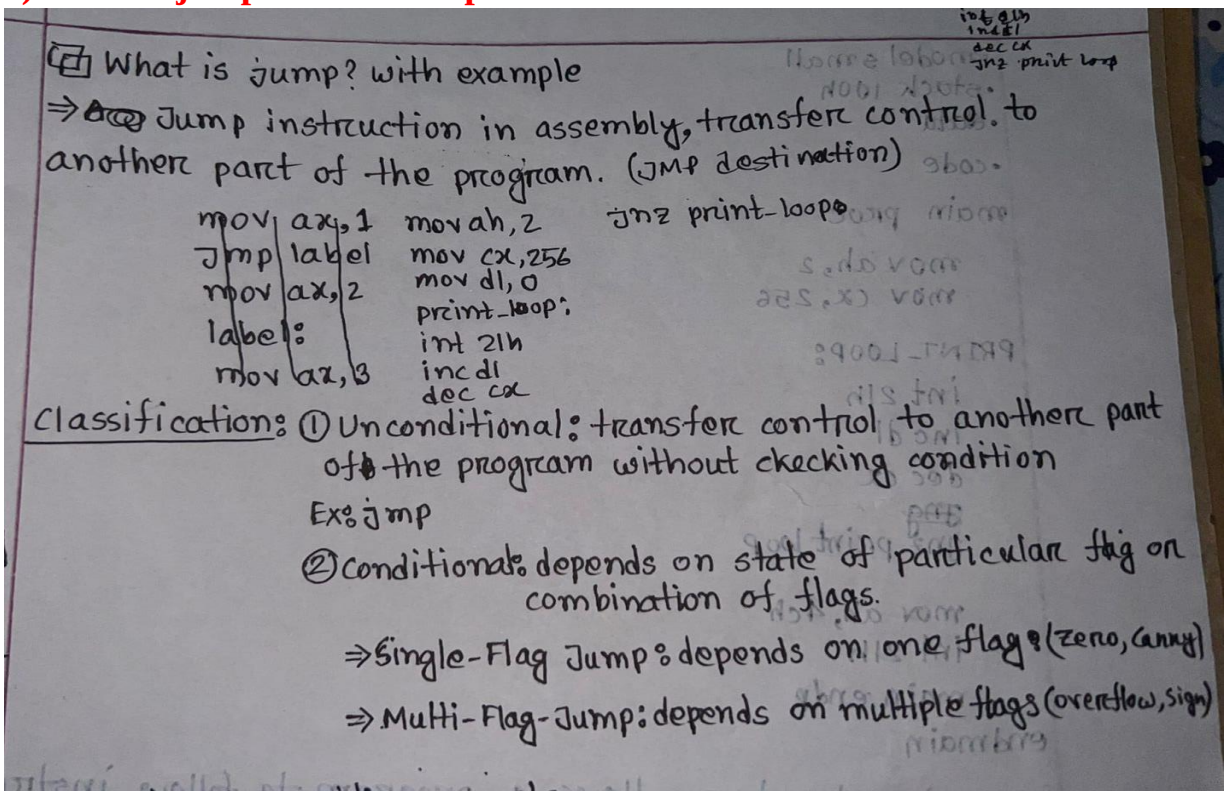
1) What is jump? With example



What is jump? with example

⇒ Jump instruction in assembly, transfer control to another part of the program. (JMP destination)

```
mov ax,1        mov ah,2        jnz print_loop
jmp label       mov cx,256
mov ax,2        mov dl,0
label:          print_loop:
                int 21h
mov ax,3        inc dl
                dec cx
```

Classification: ① Unconditional: transfer control to another part of the program without checking condition

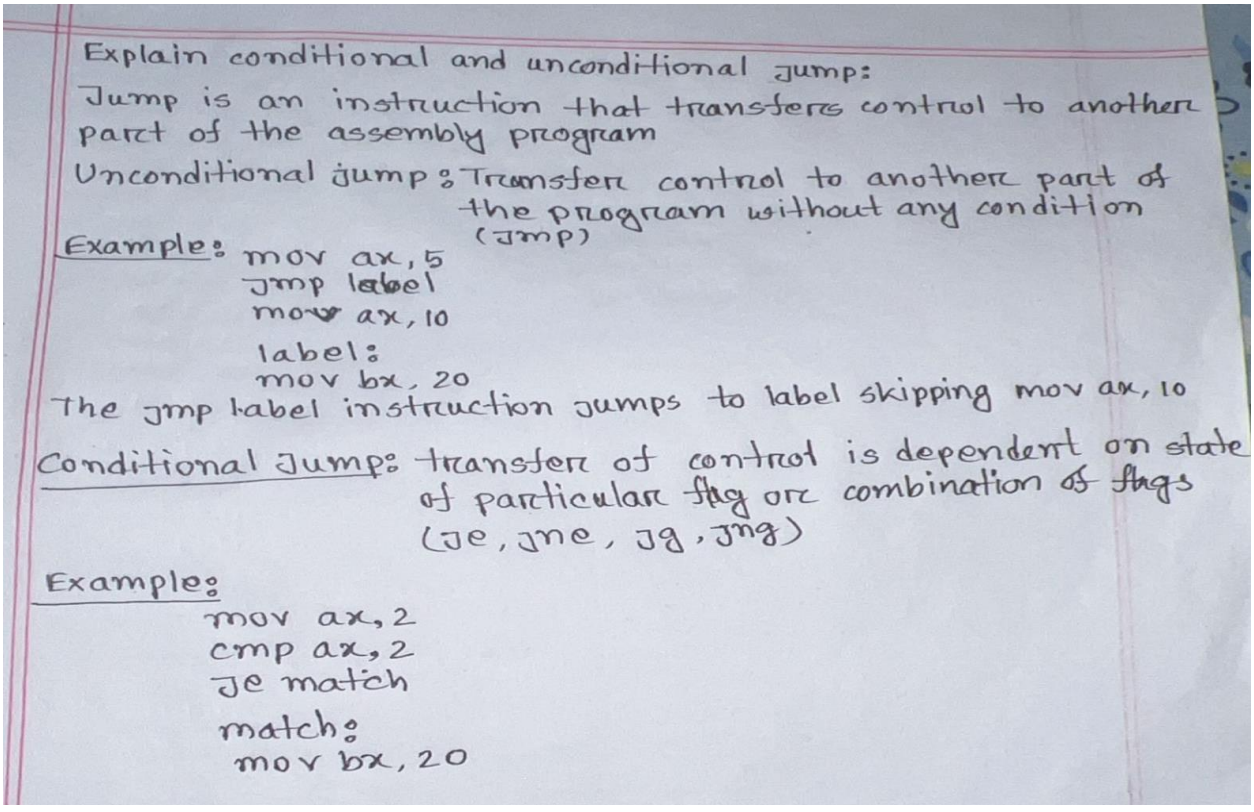
Ex: jmp

② Conditional: depends on state of particular flag or combination of flags.

⇒ Single-Flag Jump: depends on one flag: (zero, carry)

⇒ Multi-Flag-Jump: depends on multiple flags (overflow, sign)

2) Explain conditional and unconditional jump with example



Explain conditional and unconditional jump:

Jump is an instruction that transfers control to another part of the assembly program

Unconditional jump: Transfer control to another part of the program without any condition (jmp)

Example:

```
mov ax,5
jmp label
mov ax,10
label:
mov bx,20
```

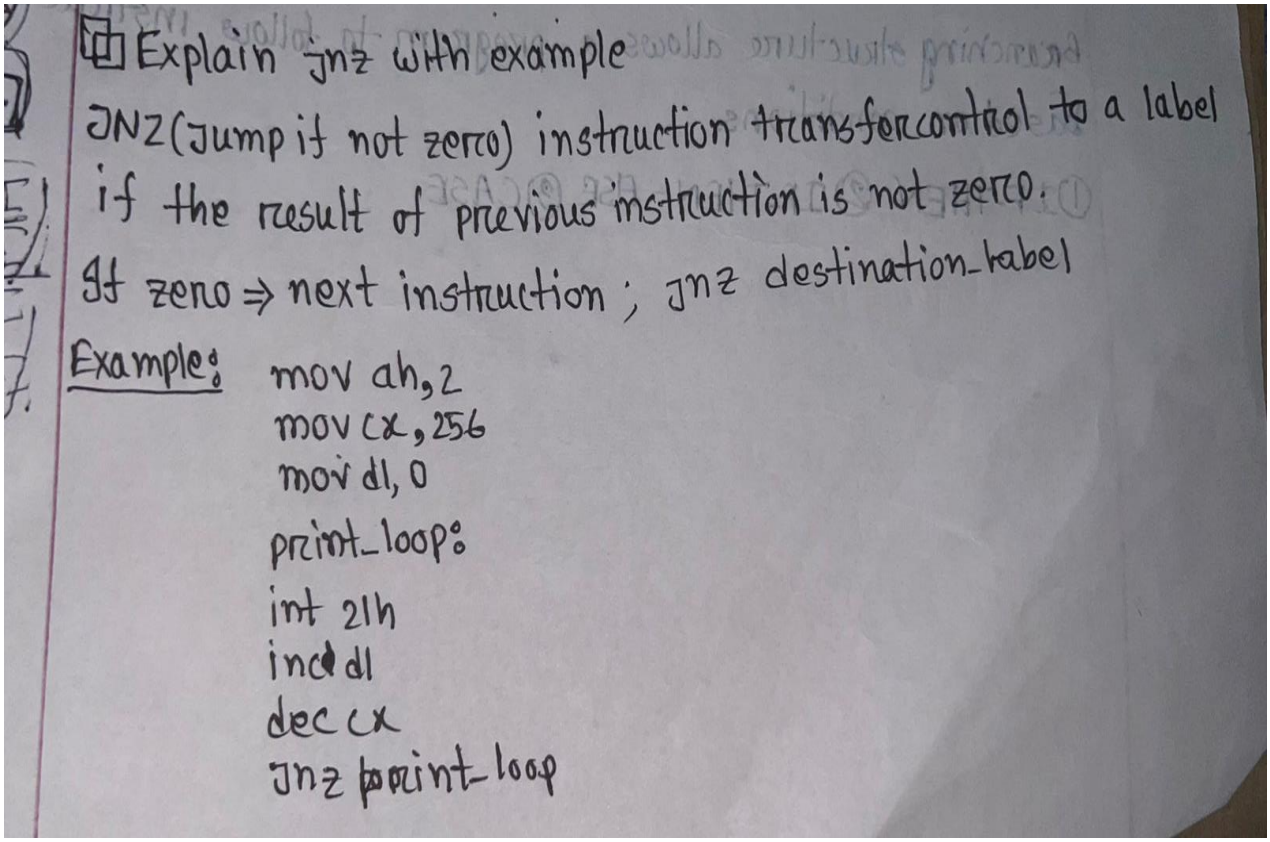
The jmp label instruction jumps to label skipping mov ax,10

Conditional Jump: transfer of control is dependent on state of particular flag or combination of flags (je, jne, jg, jng)

Example:

```
mov ax,2
cmp ax,2
je match
match:
mov bx,20
```

3) Explain JNZ with example



Explain jnz with example

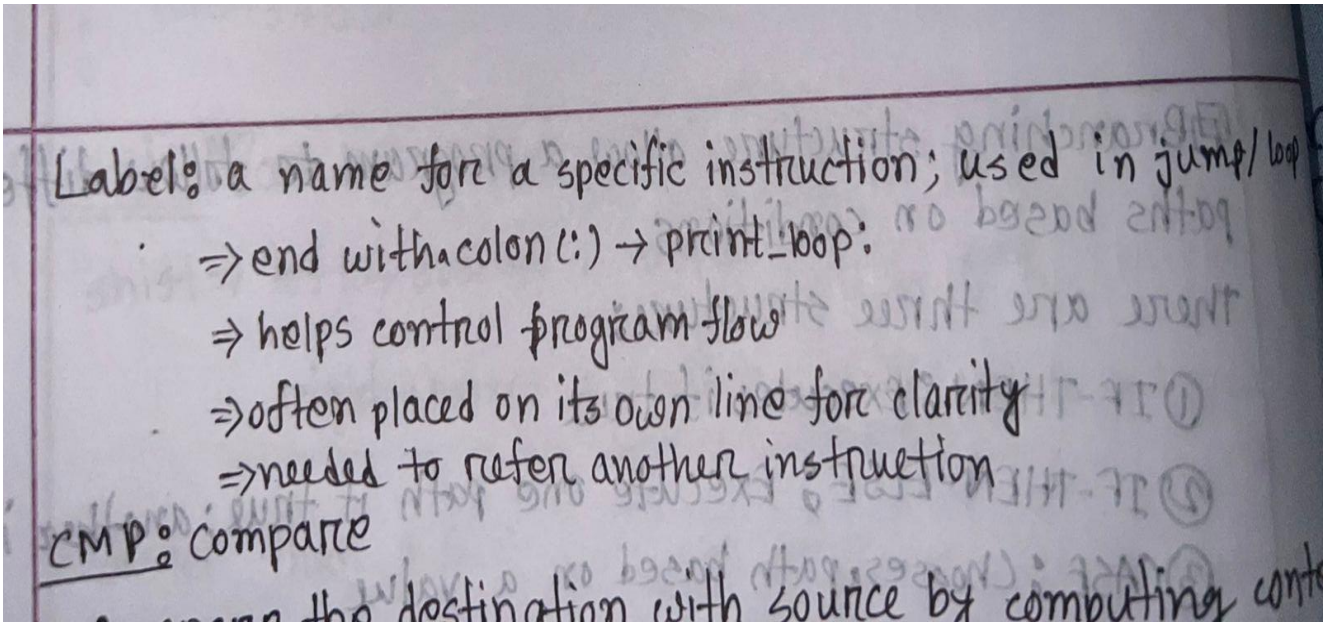
JNZ (Jump if not zero) instruction transfer control to a label if the result of previous instruction is not zero.

If zero ⇒ next instruction ; jnz destination_label

Example:

```
mov ah,2
mov cx,256
mov dl,0
print_loop:
int 21h
inc dl
dec cx
jnz print_loop
```

4) Label:



Label: a name for a specific instruction; used in jump/loop

⇒ end with a colon (:) → print_loop:

⇒ helps control program flow

⇒ often placed on its own line for clarity

⇒ needed to refer another instruction

CMP: Compare

the destination with source by computing

Signed conditional jumps:

JG JNLE	ZF=0 & SF=OF
JGE JNL	SF=OF
JL JNGE	SF<>OF
JLE JNG	ZF=1 & SF<>OF

Unsigned conditional jumps:

JA JNBE	ZF=0 & CF=0
JAE JNB	CF=0
JB JNAE	CF=1
JBE JNA	ZF=1 & CF=1

JO => CF=1 or ZF=1 ; JNO => OF=1

JC => CF=1 CF=0

5) Implementation of Conditional JUMP by CPU

 FLAGS register stores status flags from the last operation.
 Conditional jumps depend on these flags.
 If true, **IP** jumps to the target label.
 If false, **IP** stays the same, and the next instruction runs.

6) Explain CMP with example

The CMP instruction compares two values by subtracting the source from the destination, but does not store the result. It updates the FLAGS register based on the subtraction.

- CMP performs destination - source.
- The result is not stored; only the FLAGS (like Zero, Carry, Sign) are updated.
- Both operands cannot be memory locations.
- The destination cannot be a constant.

Example:

```
MOV AX, 5

MOV BX, 3

CMP AX, BX   ; Compare AX and BX

JE EQUAL_LABEL   ; Jump if equal

EQUAL_LABEL:

MOV DX, 20      ; Runs if equal
```

8) Explain JMP with example

The JMP instruction performs an **unconditional jump** to a specified destination, transferring control without conditions. It's useful to bypass range limitations of conditional jumps.

Example:

```
MOV AX, 5

JMP LABEL

MOV BX, 10

LABEL:
```

MOV CX, 15

9) JMP VS JNZ

JMP	JNZ
① Unconditional	① Conditional
② Always Jumps	② Jumps only if ZF = 0 (result≠0)
③ Used to jump regardless of state	③ Used for decision making & loops
④ jmp level ; execute without any condition	④ cmp ax, bx jnz label ; jump to label if ax = bx

10) Branching and Loop Structure

Branching structure: allow a program to follow different paths based on conditions.
There are three structure:
① IF-THEN : Executes if true
② IF-THEN-ELSE : Execute one path if true ; another if false
③ CASE : Chooses path based on a value

Looping structure : a loop is a sequence of instructions that is repeated.
① For (execute at least once)
② While (check condition at the top of loop)
③ Repeat (matter of personal preference) at least once

While vs Repeat

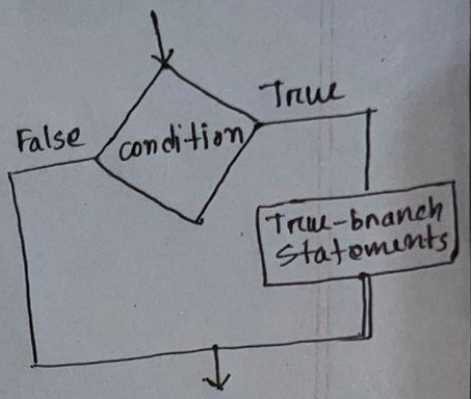
While	Repeat
condition checked before the loop	after the loop
Maybe skipped if the condition is false	Execute at least once, even if false
Require 2 jumps ; 1 conditional at top & 1 unconditional at bottom	Typically requires only one conditional jump at the end

Branching structures: It allows a program to follow different paths based on conditions

① IF-THEN (Executes if true)

If condition is true
  THEN
    execute true branch statements
END-IF

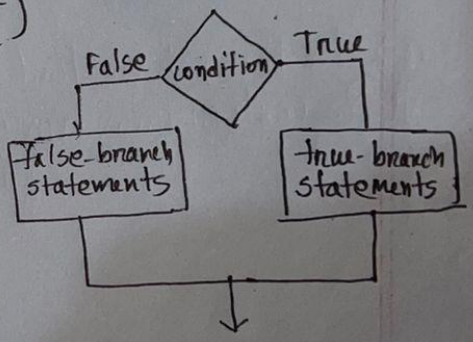
A condition is either true/false
⇒ If true, the true branch runs
⇒ If false, program continues



② IF-THEN-ELSE (Execute → one path if true / another if false)

IF condition is true
  THEN
    execute true-branch statements
  ELSE
    execute false-branch statements
END_IF

A condition is either true/false
⇒ If true, true branch runs
⇒ If false, false branch runs



③ Case (Chooses path based on values)

CASE Expression
  Values_1 : statement_1
  Values_2 : statement_2
  . . . . .
  Values_n : statement_n
END_CASE