

### Transfer of Control

- Normal statement order: sequential
- Transfer of control
- Two types: o Unconditional Transfer

o Conditional Transfer

### JMP

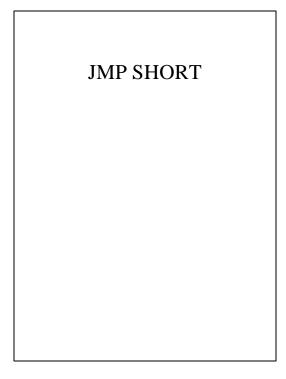
• JMP provides *unconditional transfer*.

JMP address ; unconditional xfer ;to address ;(IP gets a new ; value)

address is a user-defined label

### **JMP** Formats

• Basic forms of direct jump: JMP SHORT destination JMP NEAR PTR destination JMP FAR PTR destination



### Example

 Offset
 Machine Code Source Code

 0100
 B4 02
 start:
 mov ah, 2; loop start

 0102
 B2 41
 mov dl, 'A';
 int

 0104
 CD 21
 int
 21h
 ;disp A

 0106
 EB F8
 jmp
 start
 ;jmp back

 0108
 ..... (rest of program)
 Start
 start
 start

How does it know it's a SHORT jump?

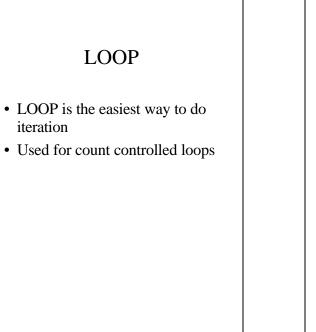
Symbol Table Symbol Value start 0100

### JMP NEAR PTR

• What if the jump in the example had been forward?

### JMP FAR PTR

- FAR PTR allows you to jump outside of your current code segment.
- This is a 5-byte instruction:
  1 for opcode
  2 for displacement (-> IP)
  2 for segment (->CS)



# LOOP, continued

- Format: LOOP destination
- What loop does: CX is the loop counter LOOP subtracts one from CX If CX is NOT equal to zero, control transfers to destination
  Example: mov cx. 5 :: cx = 5

mov ex, 5	, cx = 5
mov ax, 0	;ax = 0
start:	
add ax, 1	
loop start	jump to start;

after loop: ax = 5, cx = 0

# Another LOOP example

```
In C:

ax = 0;

for (i = 23; i >= 1; i--)

{

ax = ax + bx;

}
In Assembly:

sub ax, ax

mov cx, 23

start: add ax, bx

loop start

;jumps to start
```

# Loop: Errors to Avoid!

- Starting with CX = 0
- Altering the loop counter
- Also: Flags are not affected when LOOP decrements CX



Conditional Loops:

 o LOOPZ/LOOPE (loop if zero, loop if equal)
 o LOOPNZ/LOOPNE (loop if not zero, loop if not equal)

### LOOPZ/LOOPE

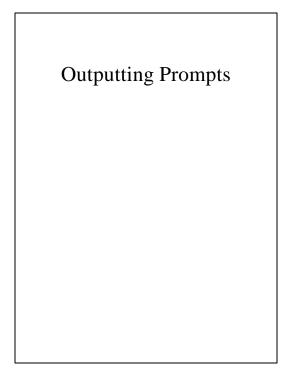
- LOOP while ZF = 1 and CX > 0
- Example, p. 203 in Irvine

### LOOPNZ/LOOPNE

- Loop while ZF = 0 and CX > 0
- Example from p. 204, Irvine

# Conditional Loop Pitfalls

- Same as for unconditional... plus:
  - o Be careful with the flags!
- Example p. 203, Irvine



### I/O in Assembly

- INT 21h is a DOS function call (DOS services)
- You saw one example of this in hw2: mov ax, 4c00h int 21h ;don't forget h!
- This returns to DOS from an executing program.
- The 4Ch that goes into AH tells DOS which function to perform (in this case, returning to DOS).

## Outputting the Prompt

• For hw3:

.data prompt1 db "Enter the amount (in cents): \$" .code .startup ;prompt for first input mov ah, 09h

- mov dx, offset prompt1
  int 21h
- What is this doing? o 09h: String output function – writes a string
  - to standard output
  - o AH = 09, DX = offset of the string.
  - o The string must be terminated by a \$ (dollar sign character)

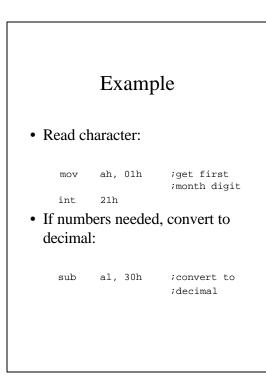
### More on Output

- If you want to output a string, followed by a carriage return and line feed (crlf), you will need to put them in your string also!
- example, p. 148 of Irvine

# Warning!

• The '\$' is important!!!

# Reading in Characters 01h – filtered input with echo waits for a single character to be entered (or, if one is in the input buffer already just grabs it) stores it in AL Input: ah = 1 Output: al = the character read o filtered? o echo?



Disadvantages of using 01h:

 o only one character at a time
 o it will read any character typed, as it is typed.
 >if you type the wrong character you can't backspace and correct it – it has already been read and processed by your program