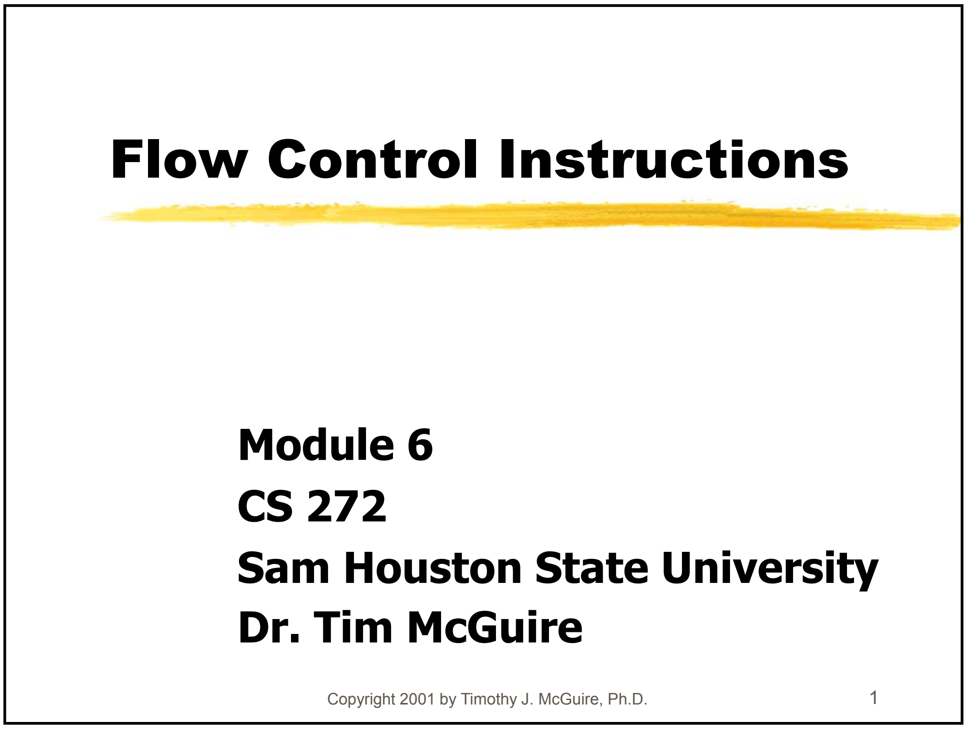
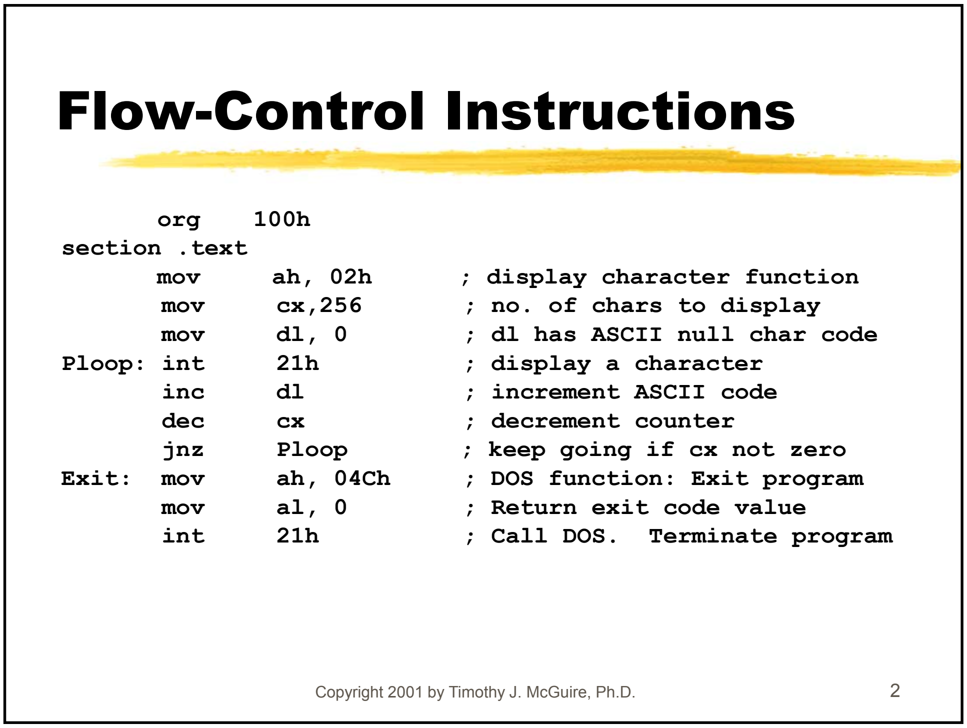


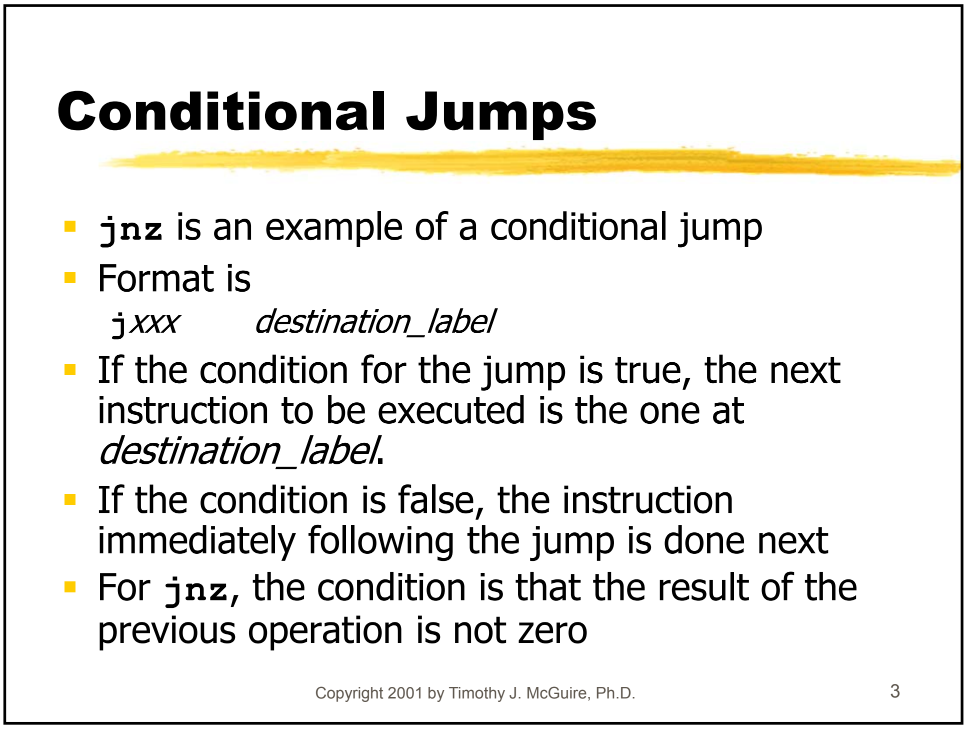
Flow Control Instructions

Module 6
CS 272
Sam Houston State University
Dr. Tim McGuire



Flow-Control Instructions

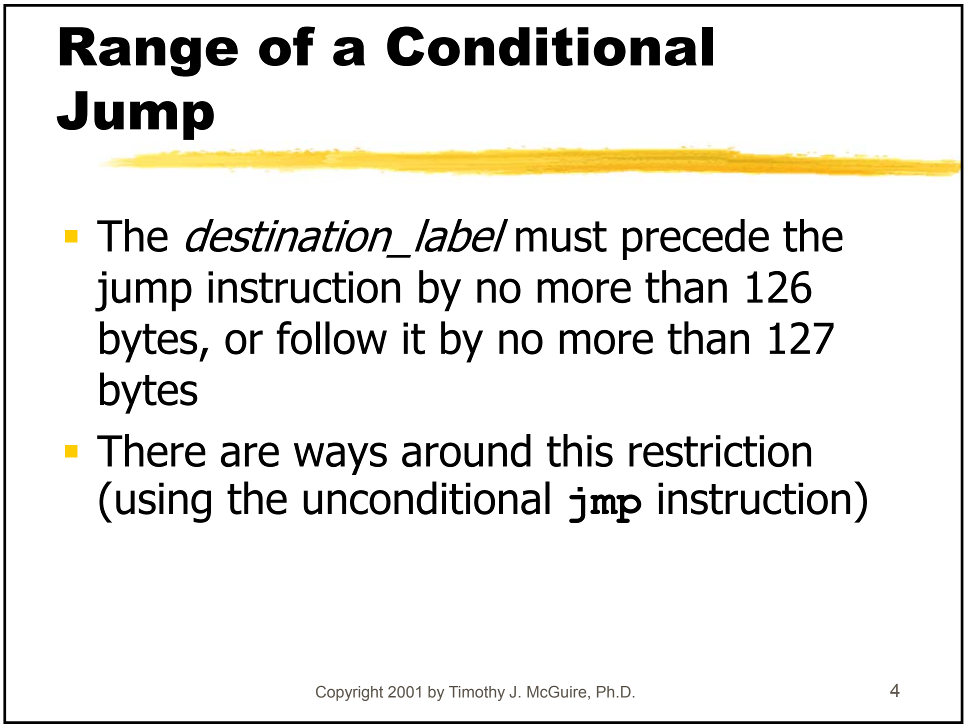
```
       org    100h
section .text
       mov     ah, 02h       ; display character function
       mov     cx,256        ; no. of chars to display
       mov     dl, 0         ; dl has ASCII null char code
Ploop: int     21h           ; display a character
       inc     dl            ; increment ASCII code
       dec     cx            ; decrement counter
       jnz     Ploop         ; keep going if cx not zero
Exit:  mov     ah, 04Ch      ; DOS function: Exit program
       mov     al, 0         ; Return exit code value
       int     21h           ; Call DOS.  Terminate program
```



Conditional Jumps

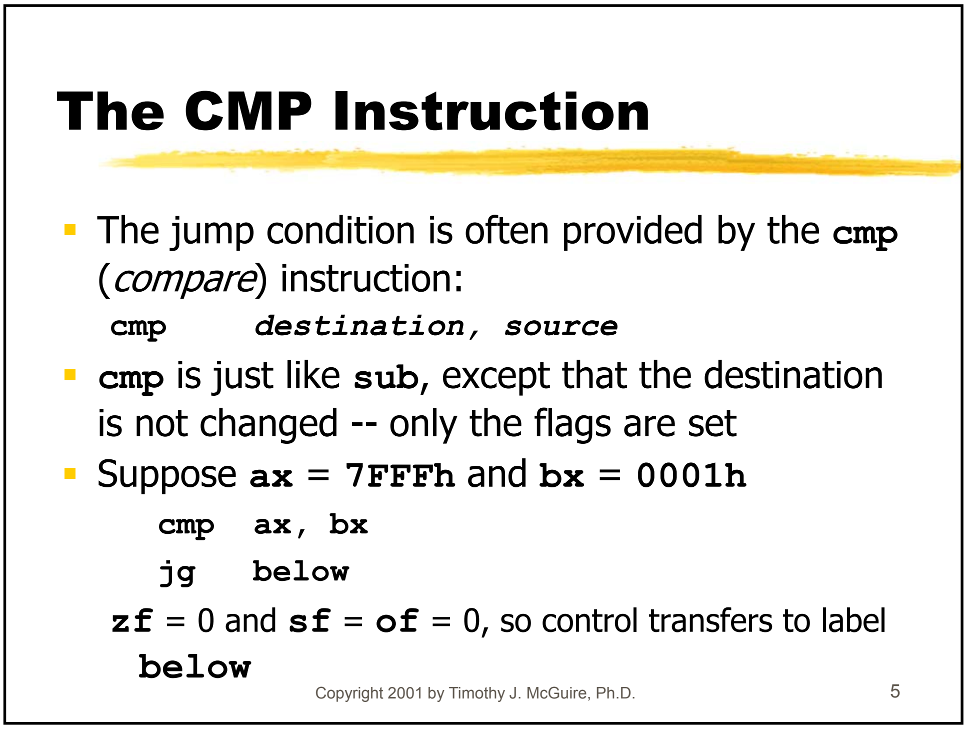
- `jnz` is an example of a conditional jump
- Format is

  `j`*xxx*   *destination_label*
- If the condition for the jump is true, the next instruction to be executed is the one at *destination_label*.
- If the condition is false, the instruction immediately following the jump is done next
- For `jnz`, the condition is that the result of the previous operation is not zero



Range of a Conditional Jump

- The *destination_label* must precede the jump instruction by no more than 126 bytes, or follow it by no more than 127 bytes
- There are ways around this restriction (using the unconditional `jmp` instruction)



The CMP Instruction

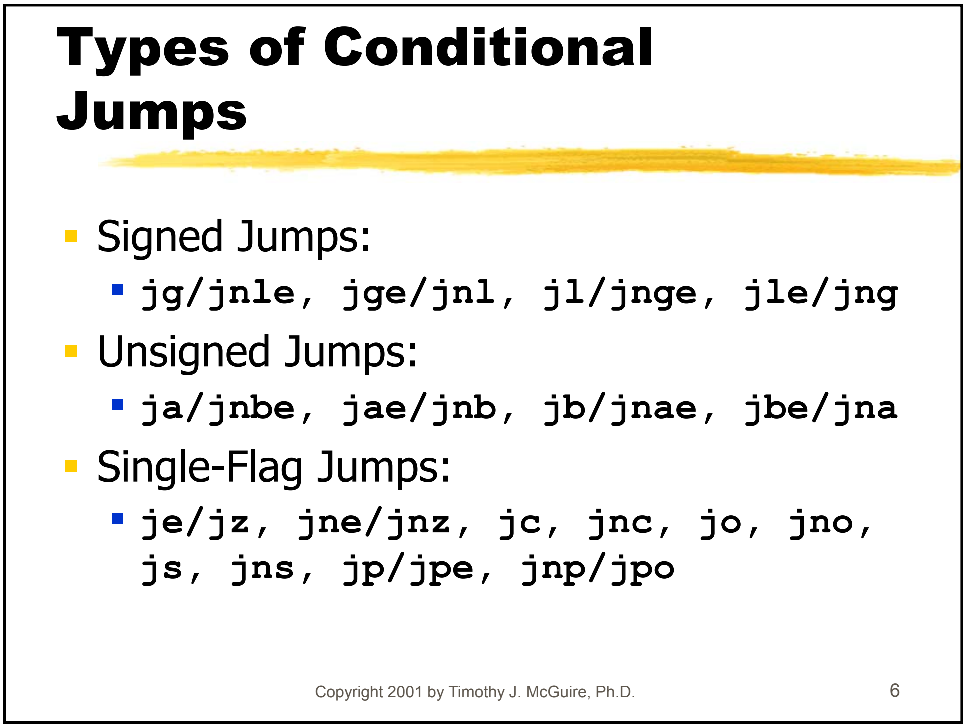
- The jump condition is often provided by the **`cmp`** (*compare*) instruction:

```
cmp     destination, source
```

- **`cmp`** is just like **`sub`**, except that the destination is not changed -- only the flags are set
- Suppose **`ax`** = **`7FFFh`** and **`bx`** = **`0001h`**

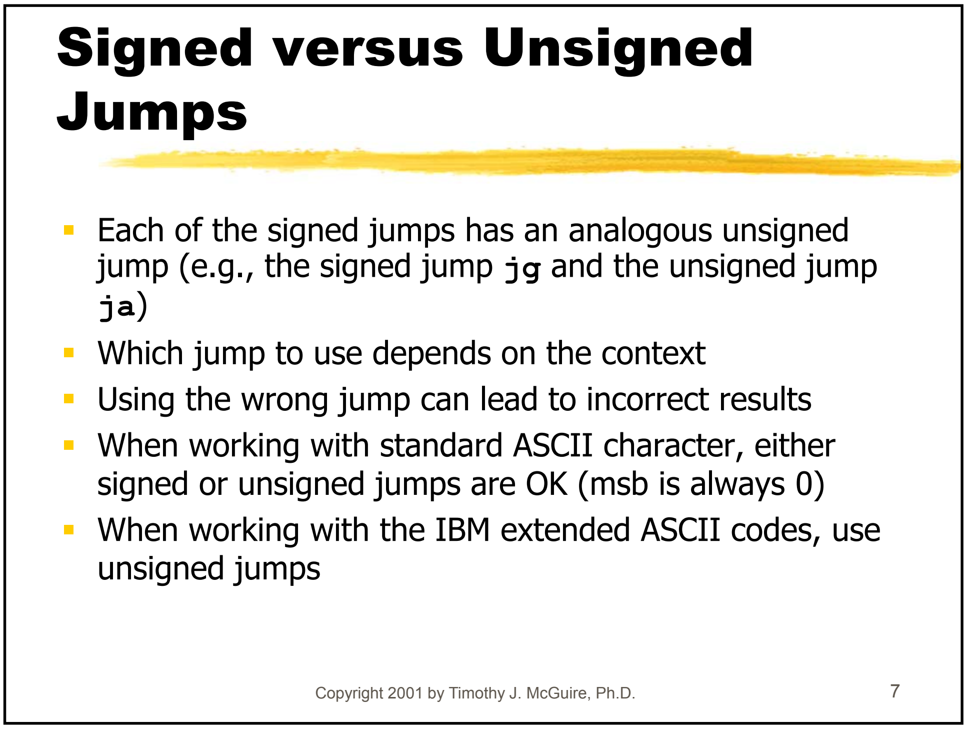
```
    cmp   ax, bx
    jg    below
```

`zf` = 0 and **`sf`** = **`of`** = 0, so control transfers to label **`below`**



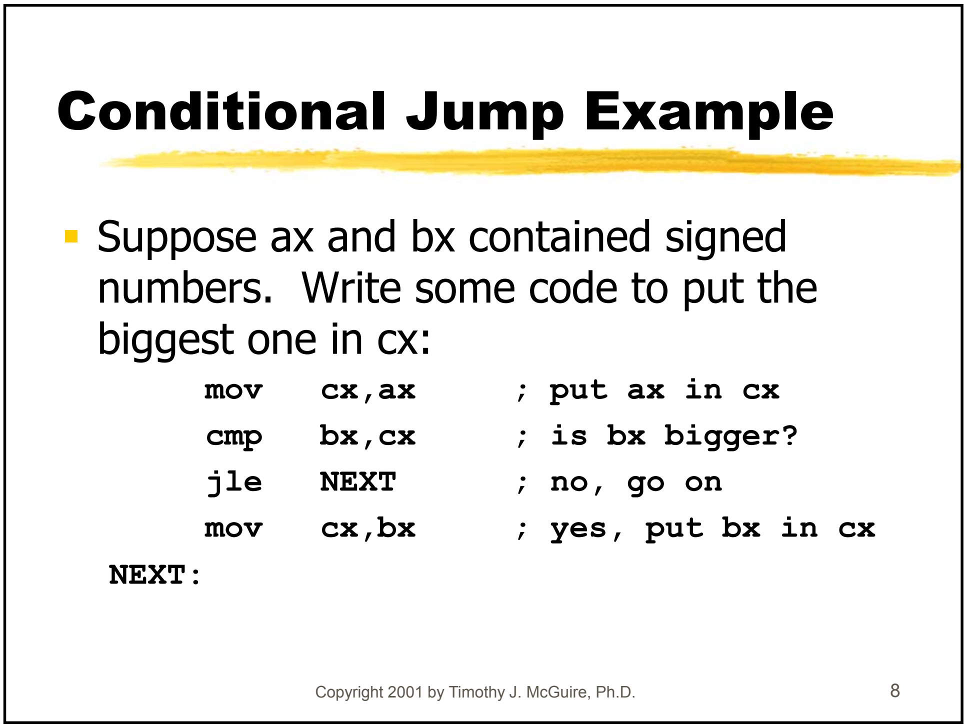
Types of Conditional Jumps

- Signed Jumps:
  - **`jg/jnle, jge/jnl, jl/jnge, jle/jng`**
- Unsigned Jumps:
  - **`ja/jnbe, jae/jnb, jb/jnae, jbe/jna`**
- Single-Flag Jumps:
  - **`je/jz, jne/jnz, jc, jnc, jo, jno, js, jns, jp/jpe, jnp/jpo`**



Signed versus Unsigned Jumps

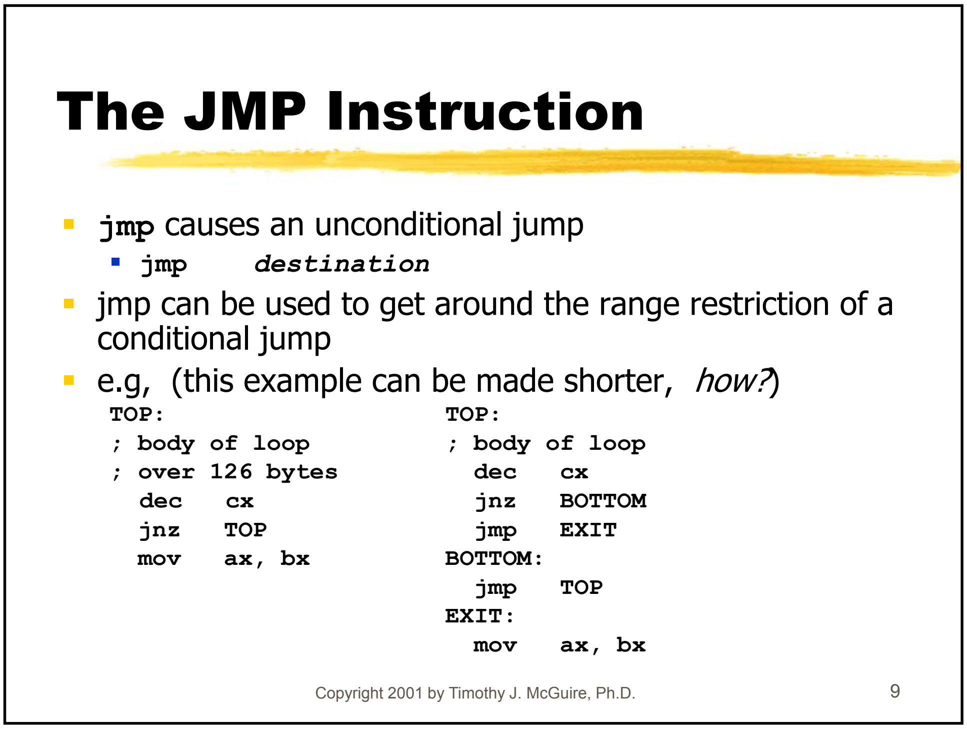
- Each of the signed jumps has an analogous unsigned jump (e.g., the signed jump `jg` and the unsigned jump `ja`)
- Which jump to use depends on the context
- Using the wrong jump can lead to incorrect results
- When working with standard ASCII character, either signed or unsigned jumps are OK (msb is always 0)
- When working with the IBM extended ASCII codes, use unsigned jumps



Conditional Jump Example

- Suppose ax and bx contained signed numbers. Write some code to put the biggest one in cx:

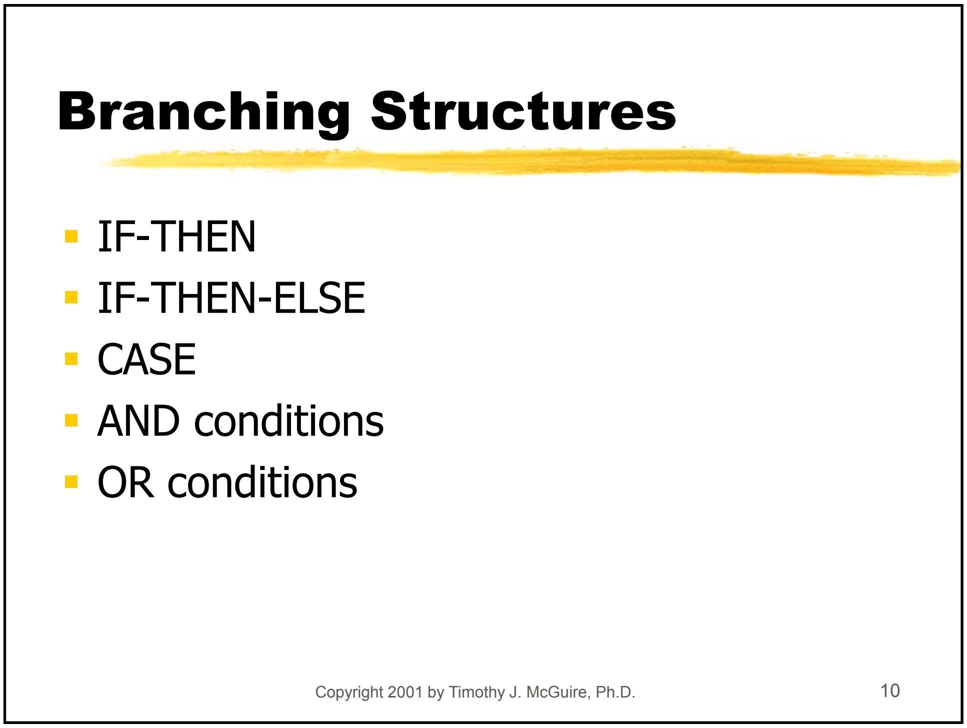
```
        mov    cx,ax      ; put ax in cx
        cmp    bx,cx      ; is bx bigger?
        jle    NEXT       ; no, go on
        mov    cx,bx      ; yes, put bx in cx
NEXT:
```



The JMP Instruction

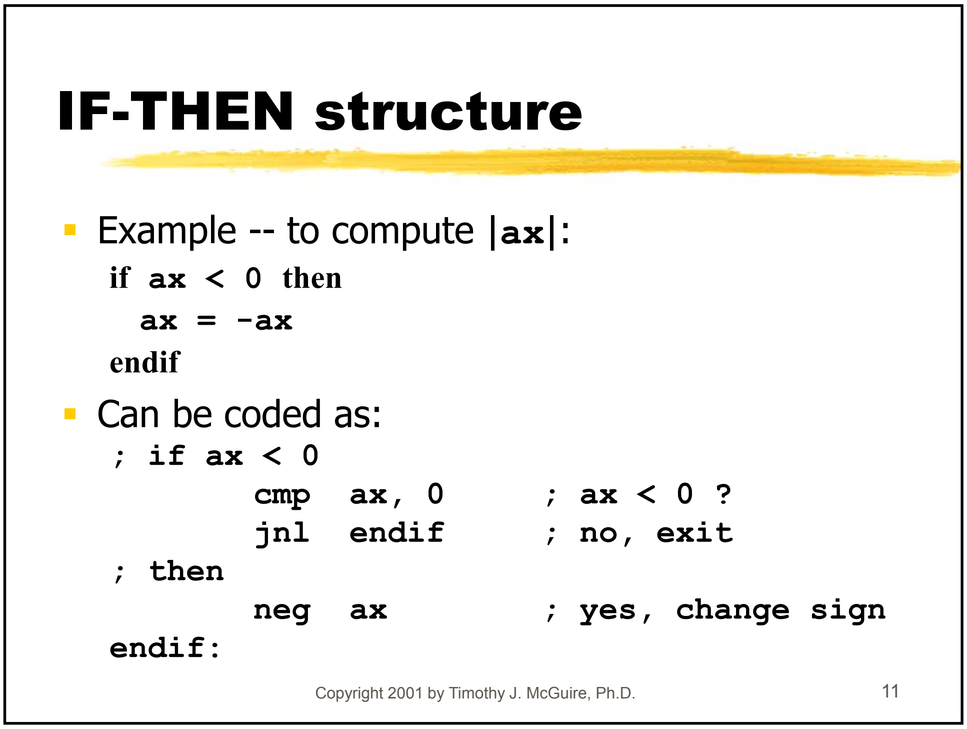
- **`jmp`** causes an unconditional jump
  - `jmp     `***`destination`***
- jmp can be used to get around the range restriction of a conditional jump
- e.g, (this example can be made shorter, *how?*)

```
TOP:                         TOP:
; body of loop               ; body of loop
; over 126 bytes               dec    cx
  dec    cx                    jnz    BOTTOM
  jnz    TOP                   jmp    EXIT
  mov    ax, bx              BOTTOM:
                               jmp    TOP
                             EXIT:
                               mov    ax, bx
```



Branching Structures

- IF-THEN
- IF-THEN-ELSE
- CASE
- AND conditions
- OR conditions



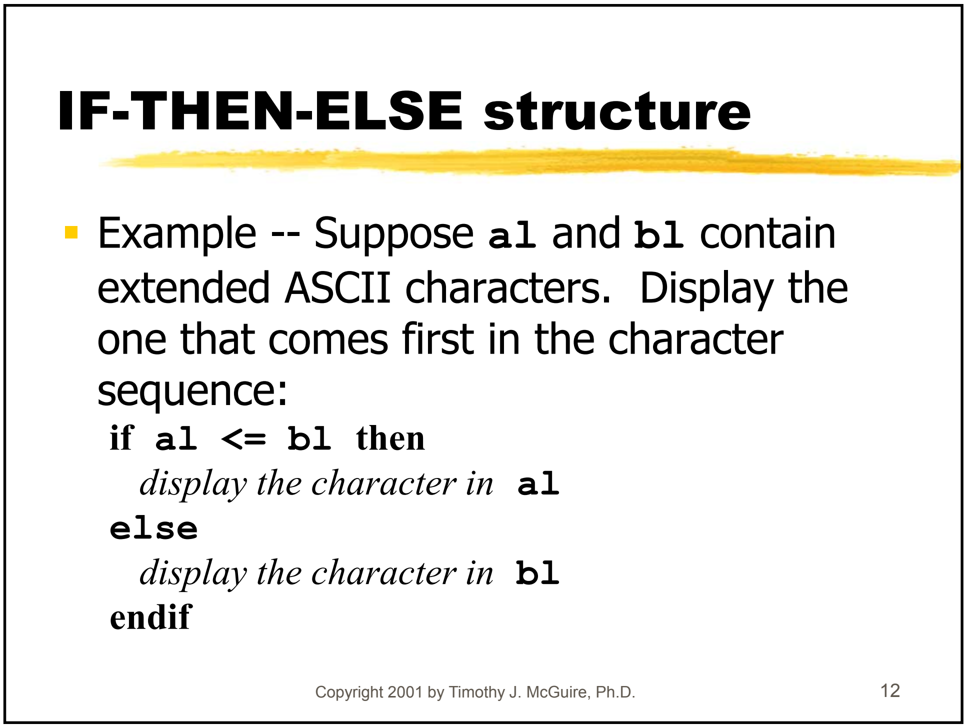
IF-THEN structure

- Example -- to compute |**ax**|:

```
if ax < 0 then
  ax = -ax
endif
```

- Can be coded as:

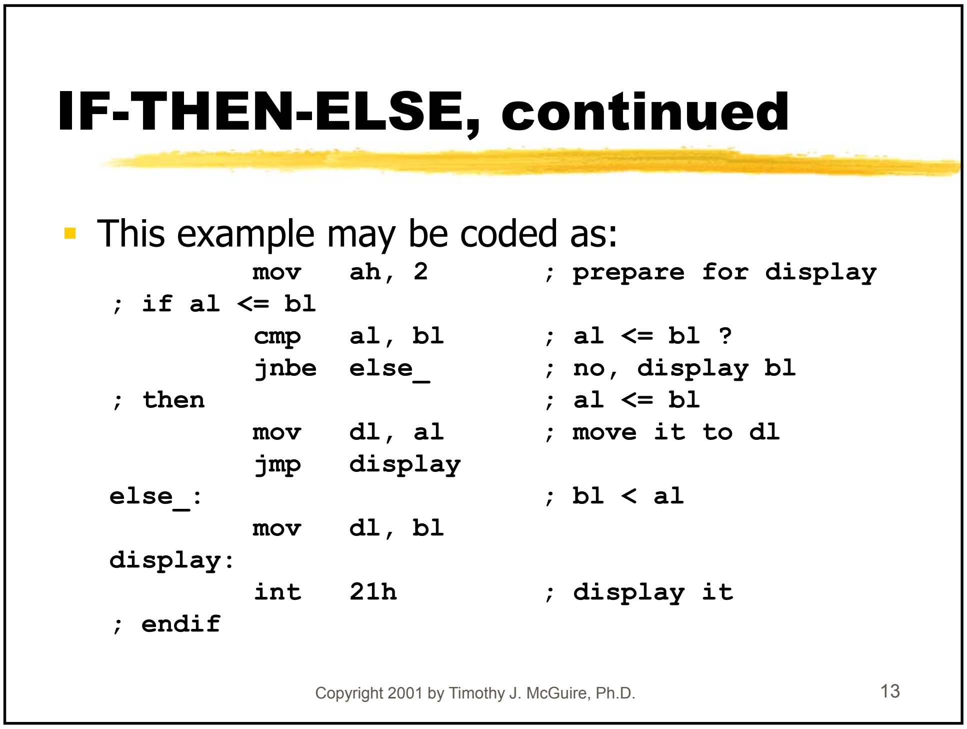
```
; if ax < 0
        cmp   ax, 0       ; ax < 0 ?
        jnl   endif       ; no, exit
; then
        neg   ax          ; yes, change sign
endif:
```



IF-THEN-ELSE structure

- Example -- Suppose `al` and `bl` contain extended ASCII characters. Display the one that comes first in the character sequence:

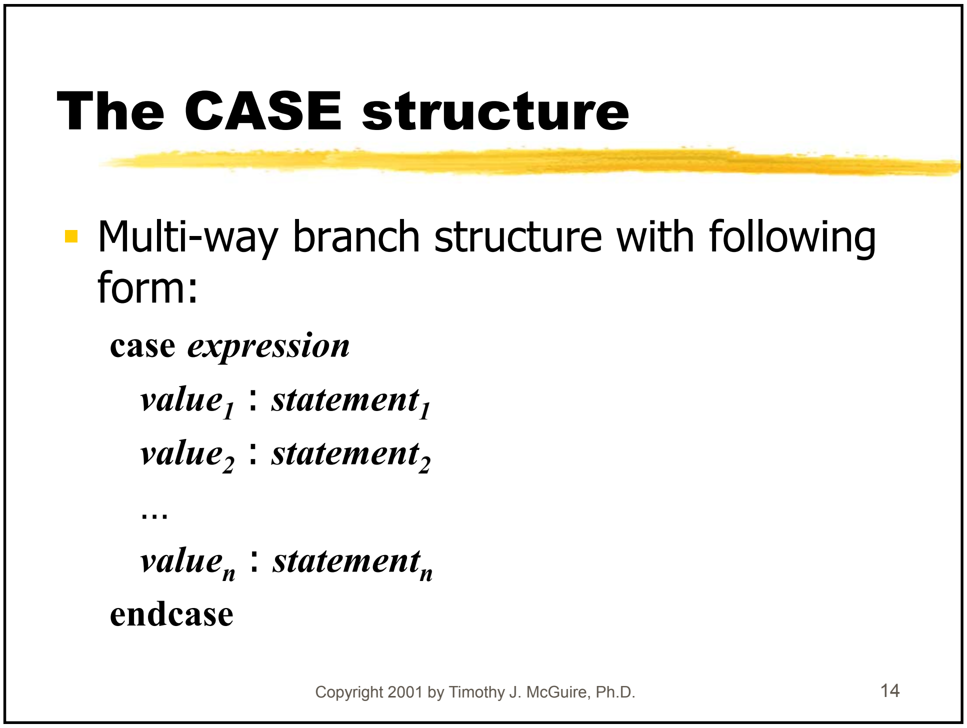
```
if al <= bl then
  display the character in al
else
  display the character in bl
endif
```



IF-THEN-ELSE, continued

- This example may be coded as:

```
          mov   ah, 2          ; prepare for display
; if al <= bl
          cmp   al, bl         ; al <= bl ?
          jnbe  else_          ; no, display bl
; then                         ; al <= bl
          mov   dl, al         ; move it to dl
          jmp   display
else_:                         ; bl < al
          mov   dl, bl
display:
          int   21h            ; display it
; endif
```



The CASE structure

- Multi-way branch structure with following form:

case ***expression***

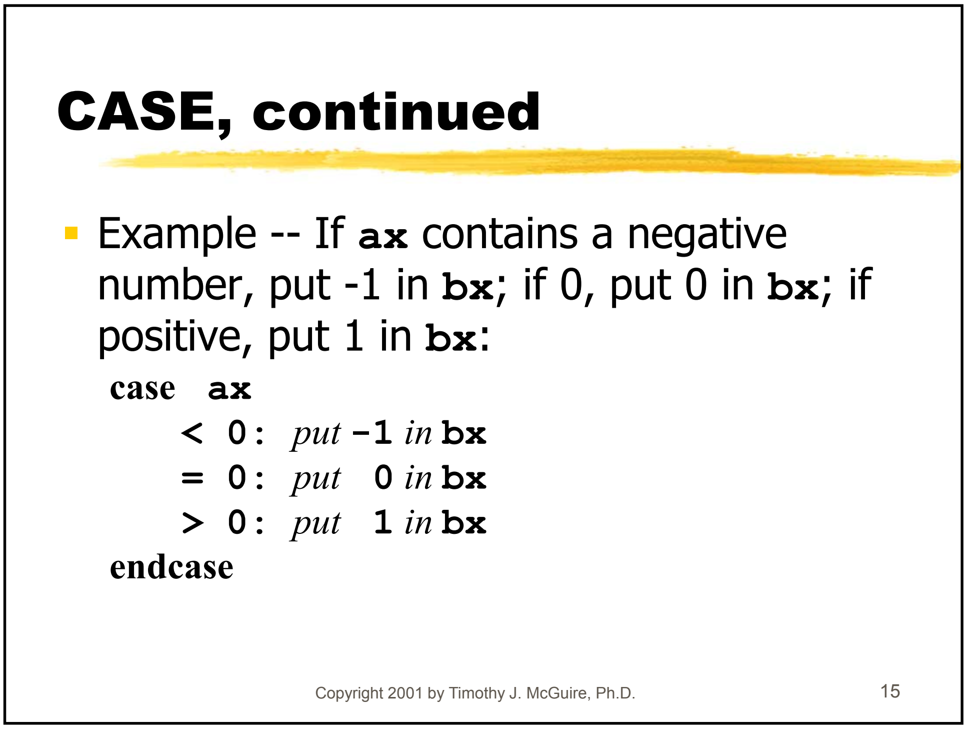
$\textit{value}_1$: $\textit{statement}_1$

$\textit{value}_2$: $\textit{statement}_2$

…

$\textit{value}_n$: $\textit{statement}_n$

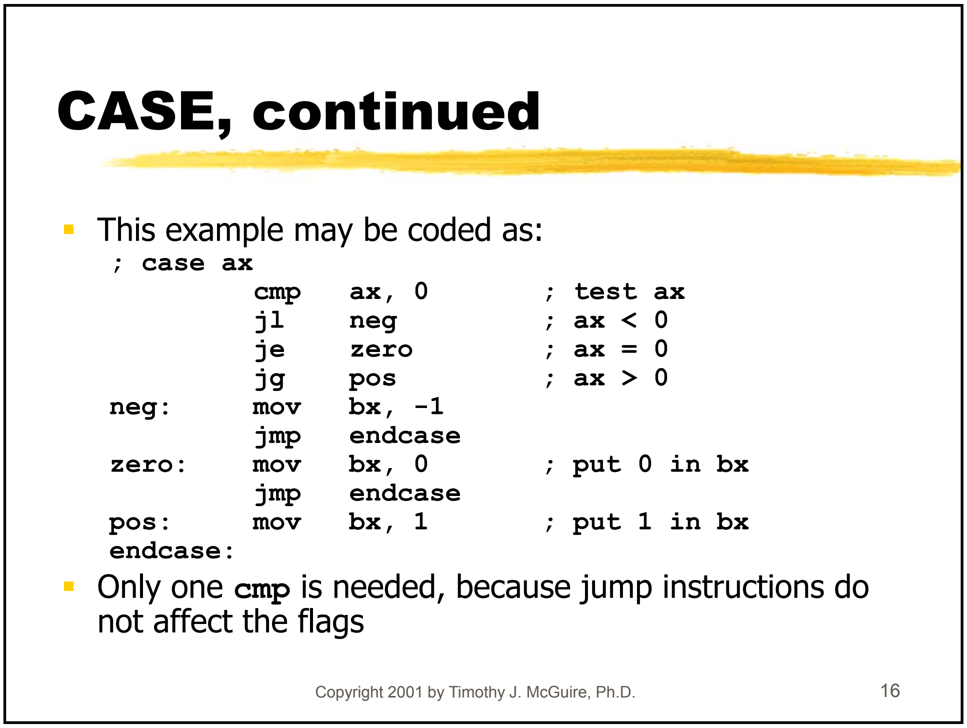
endcase



CASE, continued

- Example -- If **`ax`** contains a negative number, put -1 in **`bx`**; if 0, put 0 in **`bx`**; if positive, put 1 in **`bx`**:

```
case   ax
     < 0:  put -1 in bx
     = 0:  put  0 in bx
     > 0:  put  1 in bx
endcase
```



CASE, continued

- This example may be coded as:

```
; case ax
         cmp   ax, 0        ; test ax
         jl    neg          ; ax < 0
         je    zero         ; ax = 0
         jg    pos          ; ax > 0
neg:     mov   bx, -1
         jmp   endcase
zero:    mov   bx, 0        ; put 0 in bx
         jmp   endcase
pos:     mov   bx, 1        ; put 1 in bx
endcase:
```

- Only one **`cmp`** is needed, because jump instructions do not affect the flags

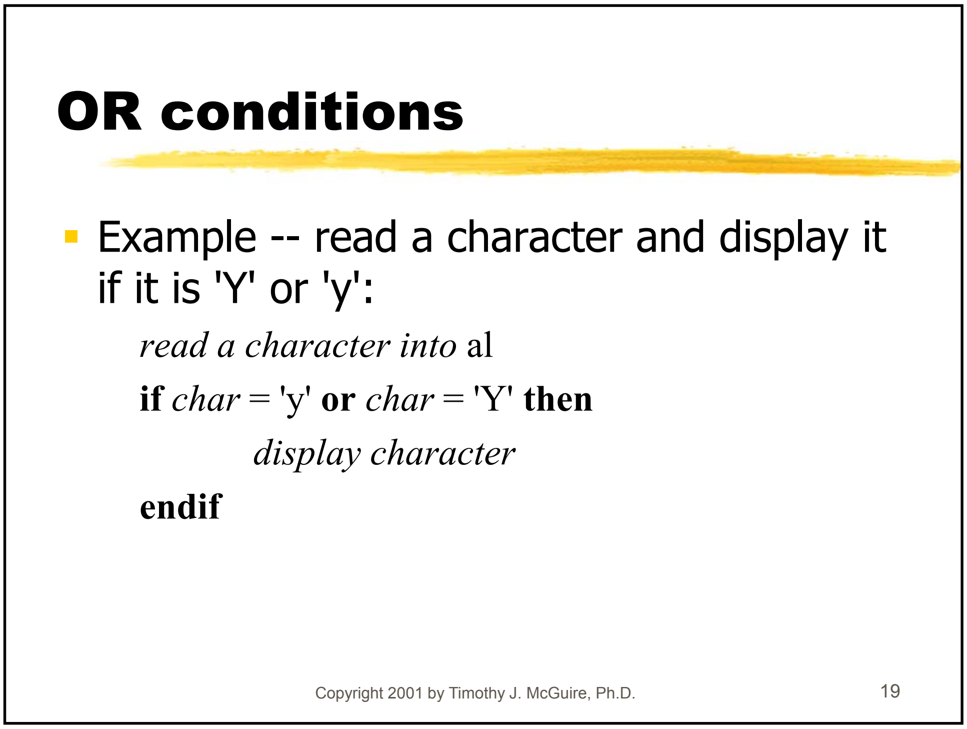
AND conditions

- Example -- read a character and display it if it is uppercase:

```
read a character into al
if char >= 'A' and char <= 'Z' then
        display character
endif
```

AND conditions, continued

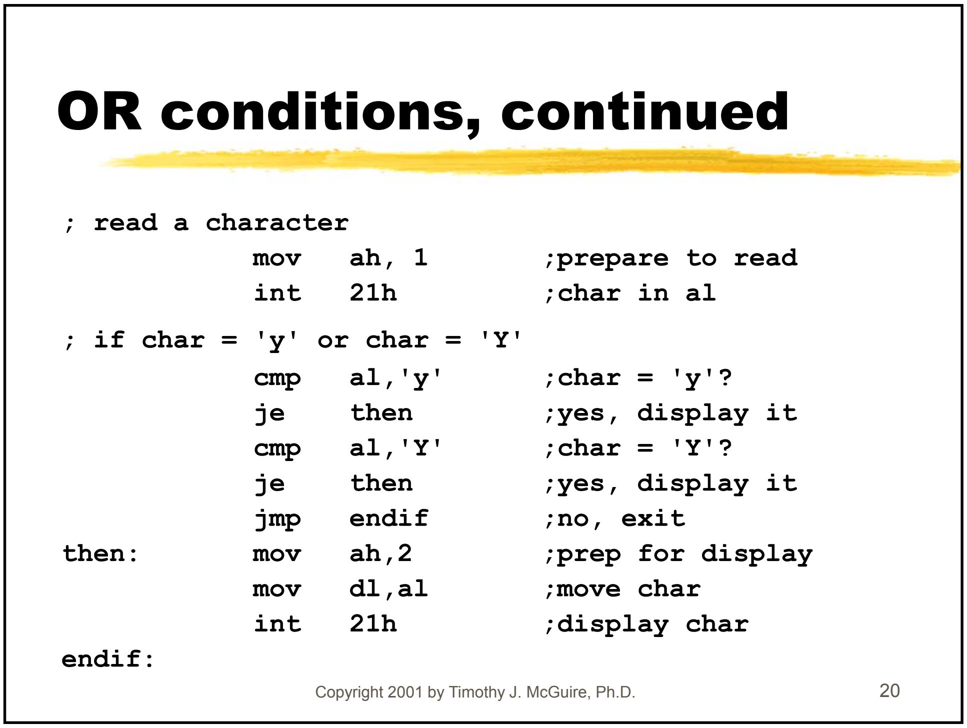
```
; read a character
            mov   ah, 1        ;prepare to read
            int   21h          ;char in al
; if char >= 'A' and char <= 'Z'
            cmp   al,'A'       ;char >= 'A'?
            jnge  endif        ;no, exit
            cmp   al,'Z'       ;char <= 'Z'?
            jnle  endif        ;no, exit
;then display character
            mov   dl,al        ;get char
            mov   ah,2         ;prep for display
            int   21h          ;display char
endif:
```



OR conditions

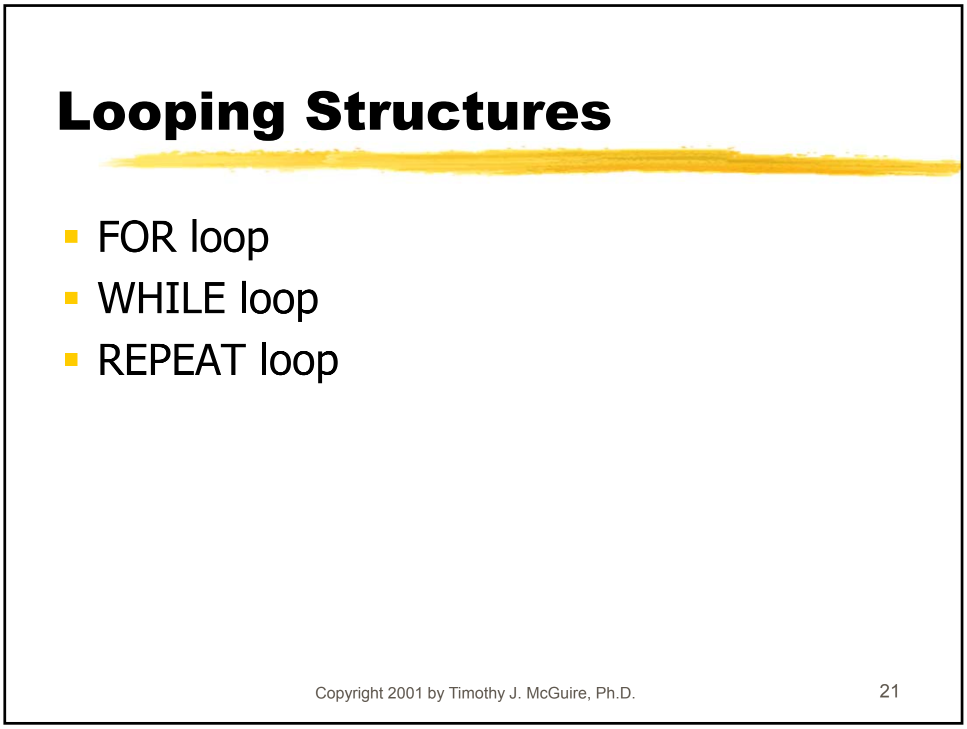
- Example -- read a character and display it if it is 'Y' or 'y':

```
read a character into al
if char = 'y' or char = 'Y' then
        display character
endif
```



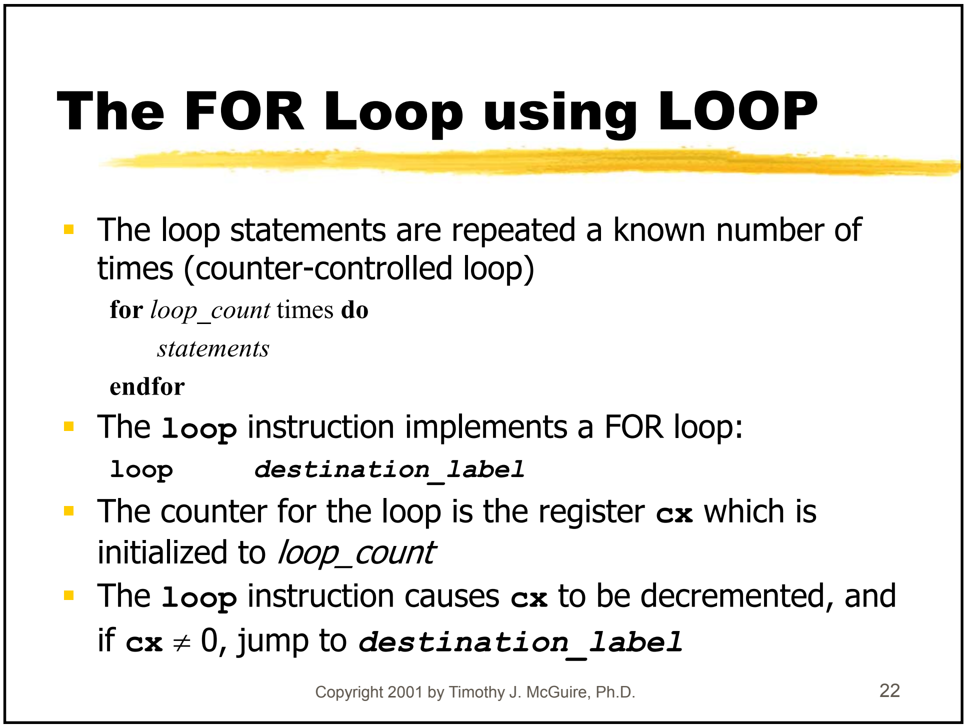
OR conditions, continued

```
; read a character
            mov   ah, 1         ;prepare to read
            int   21h           ;char in al
; if char = 'y' or char = 'Y'
            cmp   al,'y'        ;char = 'y'?
            je    then          ;yes, display it
            cmp   al,'Y'        ;char = 'Y'?
            je    then          ;yes, display it
            jmp   endif         ;no, exit
then:       mov   ah,2          ;prep for display
            mov   dl,al         ;move char
            int   21h           ;display char
endif:
```



Looping Structures

- FOR loop
- WHILE loop
- REPEAT loop



The FOR Loop using LOOP

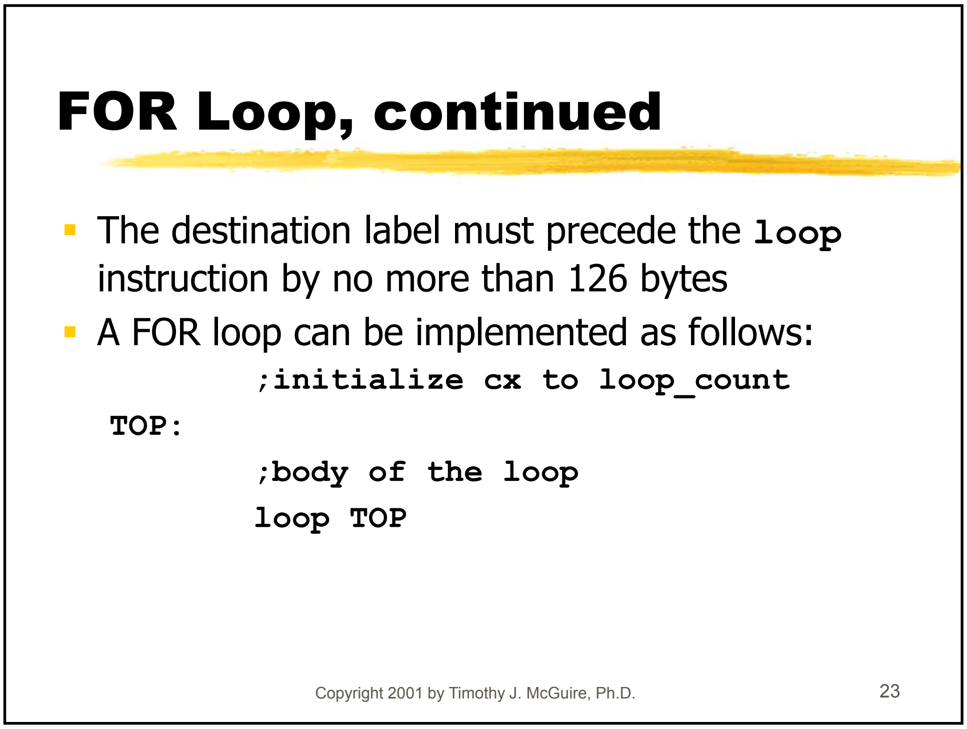
- The loop statements are repeated a known number of times (counter-controlled loop)

```
for loop_count times do
    statements
endfor
```

- The **`loop`** instruction implements a FOR loop:

```
loop     destination_label
```

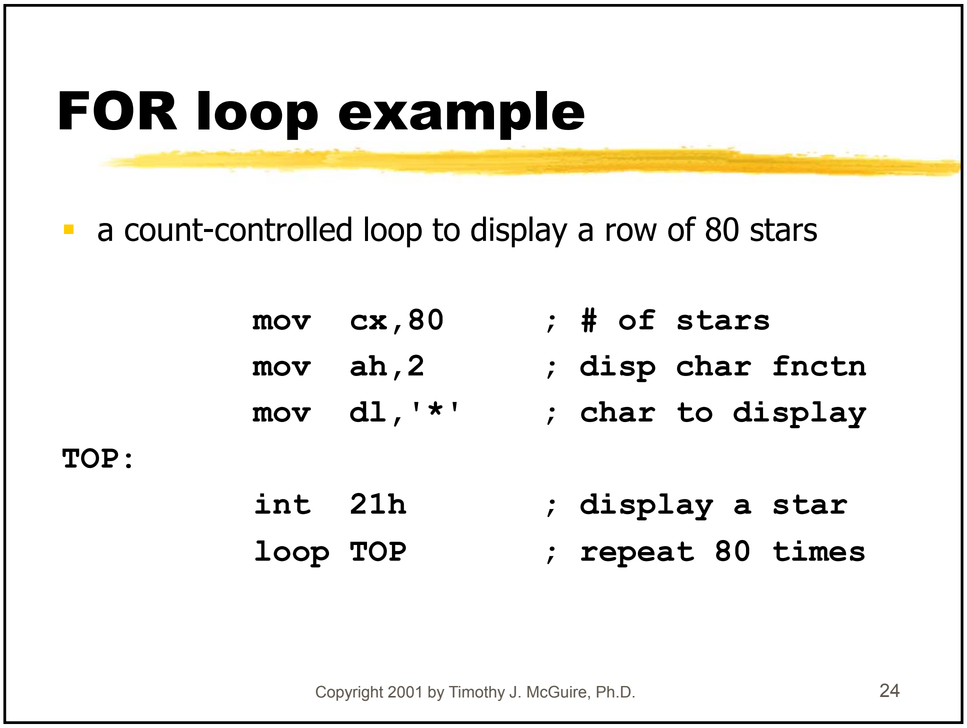
- The counter for the loop is the register **`cx`** which is initialized to *loop_count*
- The **`loop`** instruction causes **`cx`** to be decremented, and if **`cx`** ≠ 0, jump to ***`destination_label`***



FOR Loop, continued

- The destination label must precede the **`loop`** instruction by no more than 126 bytes
- A FOR loop can be implemented as follows:

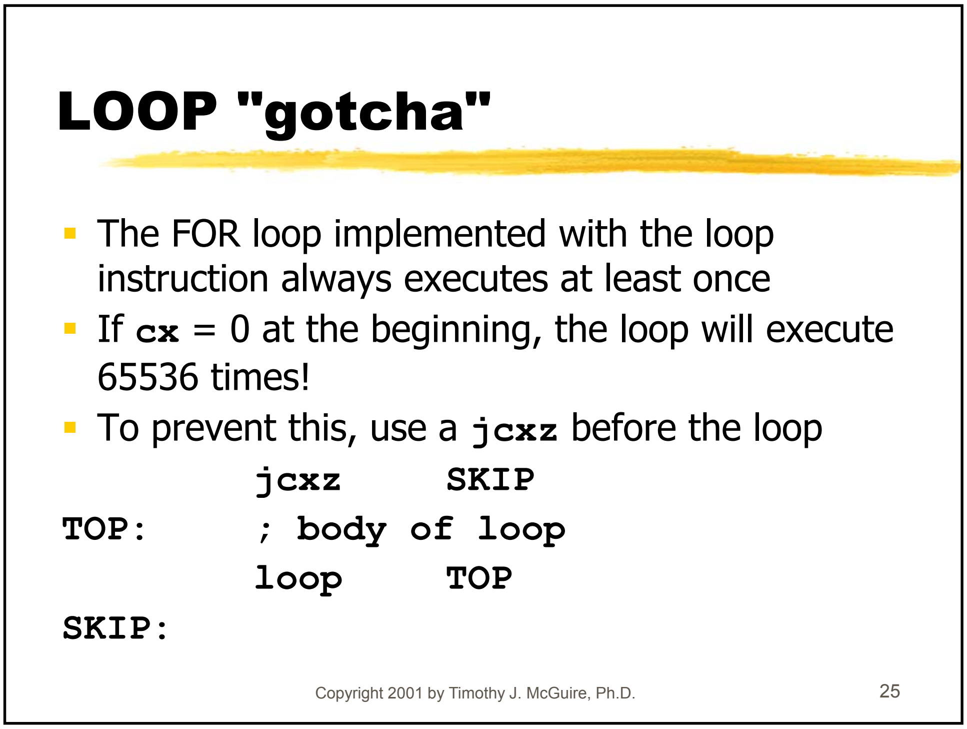
```
            ;initialize cx to loop_count
    TOP:
            ;body of the loop
            loop TOP
```



FOR loop example

- a count-controlled loop to display a row of 80 stars

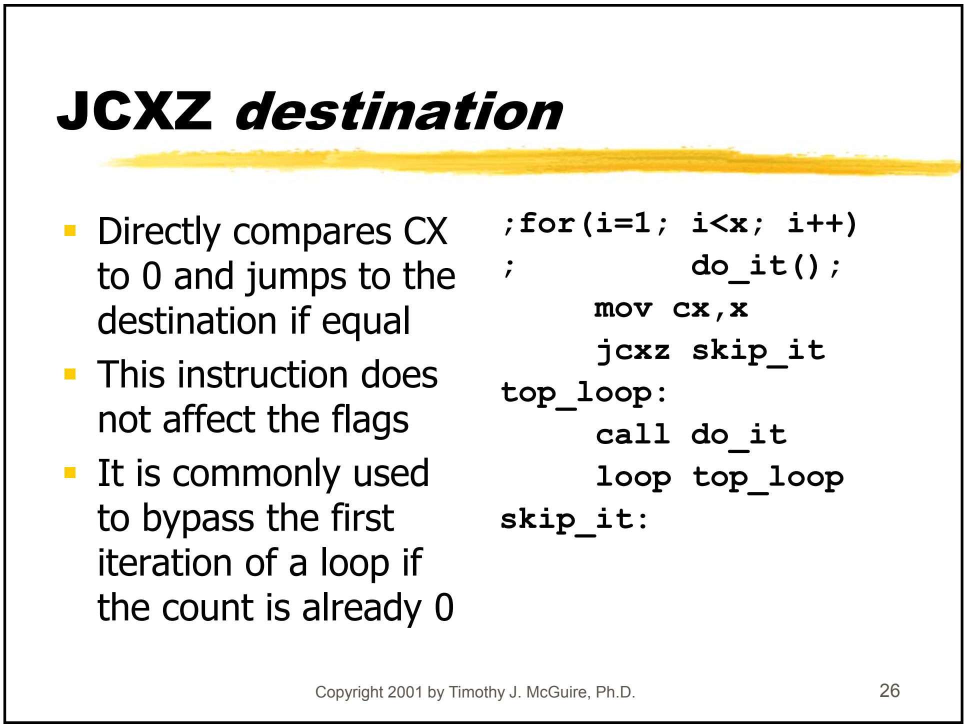
```
            mov   cx,80      ; # of stars
            mov   ah,2       ; disp char fnctn
            mov   dl,'*'     ; char to display
TOP:
            int   21h        ; display a star
            loop TOP         ; repeat 80 times
```



LOOP "gotcha"

- The FOR loop implemented with the loop instruction always executes at least once
- If **`cx`** = 0 at the beginning, the loop will execute 65536 times!
- To prevent this, use a **`jcxz`** before the loop

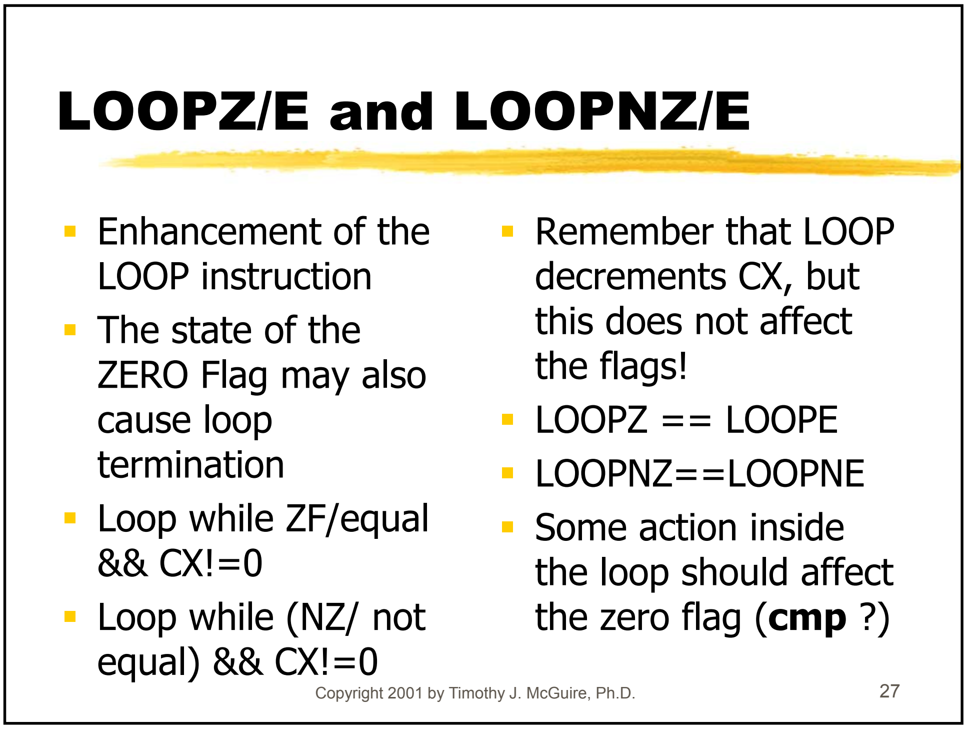
```
          jcxz     SKIP
TOP:      ; body of loop
          loop     TOP
SKIP:
```



JCXZ *destination*

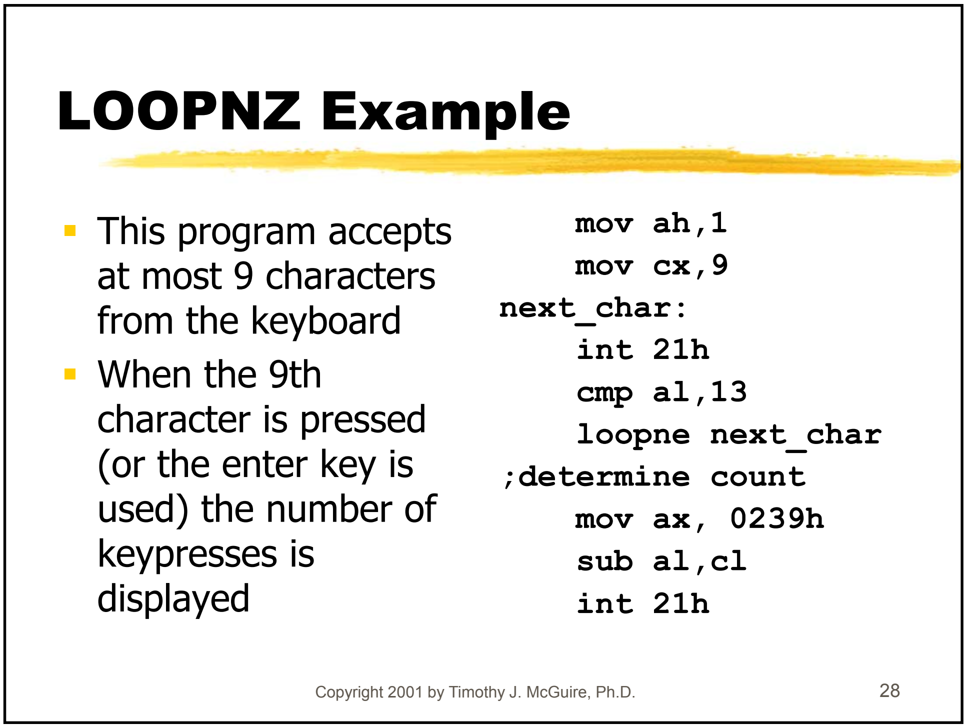
- Directly compares CX to 0 and jumps to the destination if equal
- This instruction does not affect the flags
- It is commonly used to bypass the first iteration of a loop if the count is already 0

```
;for(i=1; i<x; i++)
;         do_it();
     mov cx,x
     jcxz skip_it
top_loop:
     call do_it
     loop top_loop
skip_it:
```



LOOPZ/E and LOOPNZ/E

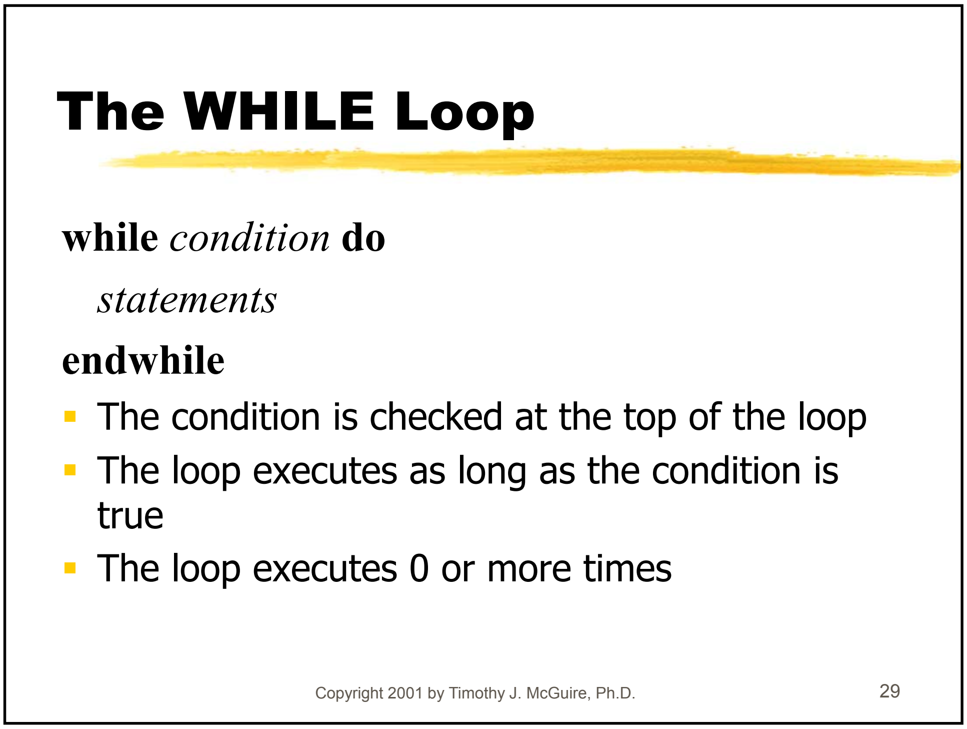
- Enhancement of the LOOP instruction
- The state of the ZERO Flag may also cause loop termination
- Loop while ZF/equal && CX!=0
- Loop while (NZ/ not equal) && CX!=0
- Remember that LOOP decrements CX, but this does not affect the flags!
- LOOPZ == LOOPE
- LOOPNZ==LOOPNE
- Some action inside the loop should affect the zero flag (**cmp** ?)



LOOPNZ Example

- This program accepts at most 9 characters from the keyboard
- When the 9th character is pressed (or the enter key is used) the number of keypresses is displayed

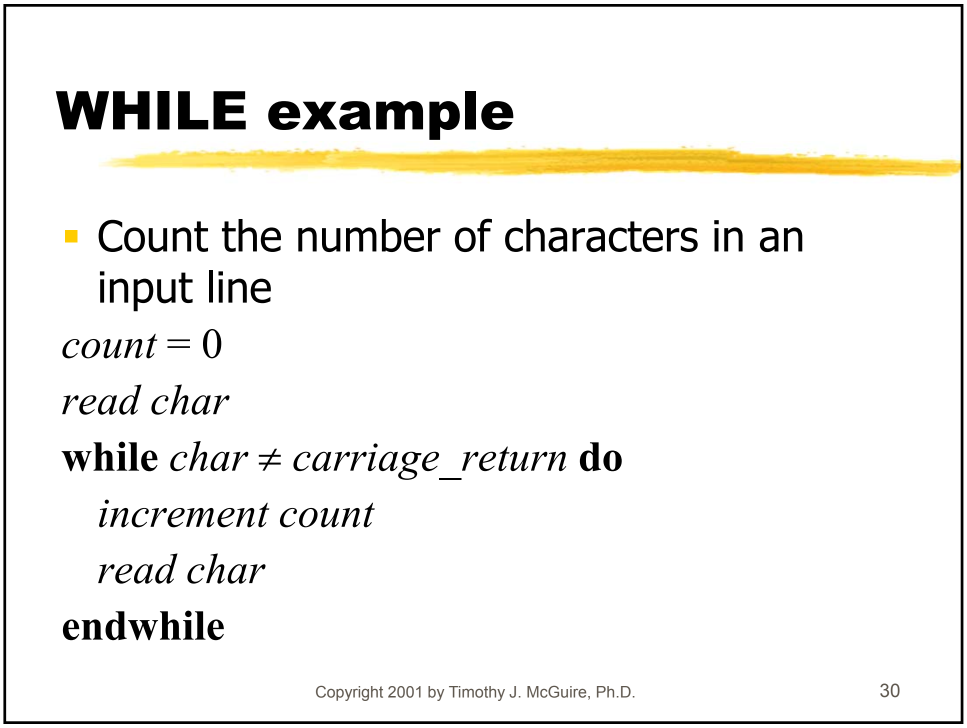
```
    mov ah,1
    mov cx,9
next_char:
    int 21h
    cmp al,13
    loopne next_char
;determine count
    mov ax, 0239h
    sub al,cl
    int 21h
```



The WHILE Loop

```
while condition do
   statements
endwhile
```

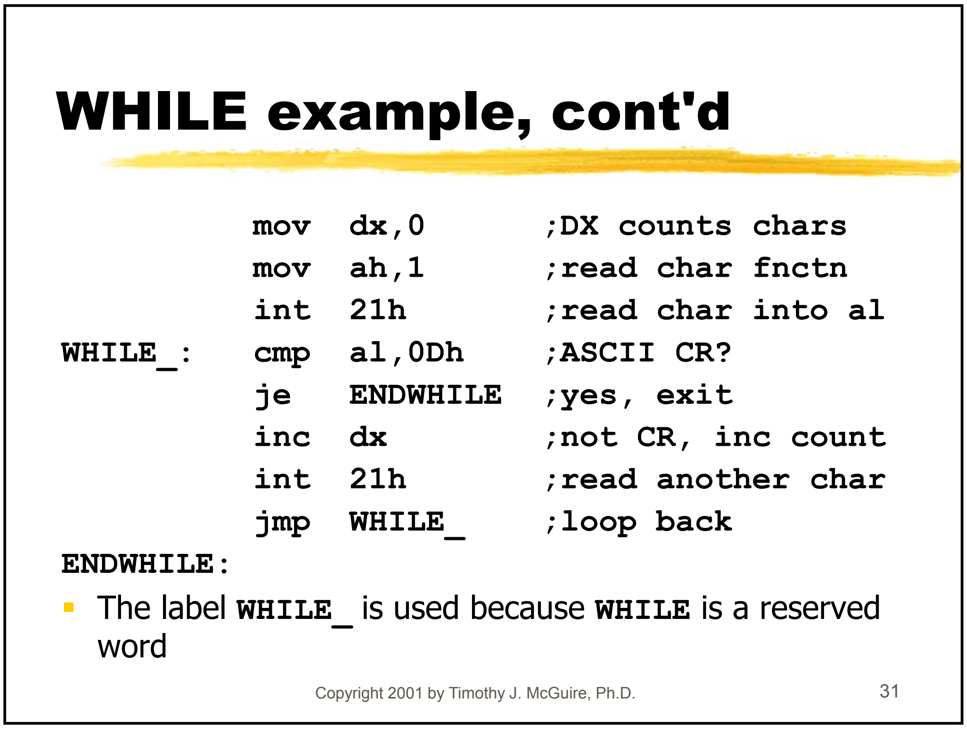
- The condition is checked at the top of the loop
- The loop executes as long as the condition is true
- The loop executes 0 or more times



WHILE example

- Count the number of characters in an input line

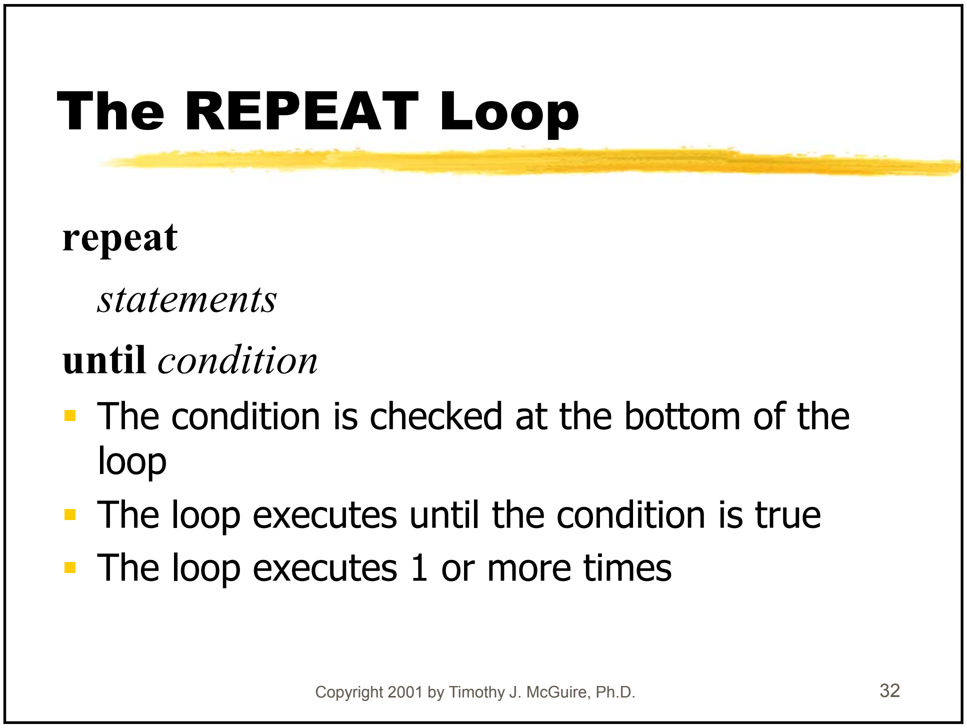
```
count = 0
read char
while char ≠ carriage_return do
   increment count
   read char
endwhile
```



WHILE example, cont'd

```
          mov   dx,0       ;DX counts chars
          mov   ah,1       ;read char fnctn
          int   21h        ;read char into al
WHILE_:   cmp   al,0Dh     ;ASCII CR?
          je    ENDWHILE   ;yes, exit
          inc   dx         ;not CR, inc count
          int   21h        ;read another char
          jmp   WHILE_     ;loop back
ENDWHILE:
```

- The label **`WHILE_`** is used because **`WHILE`** is a reserved word



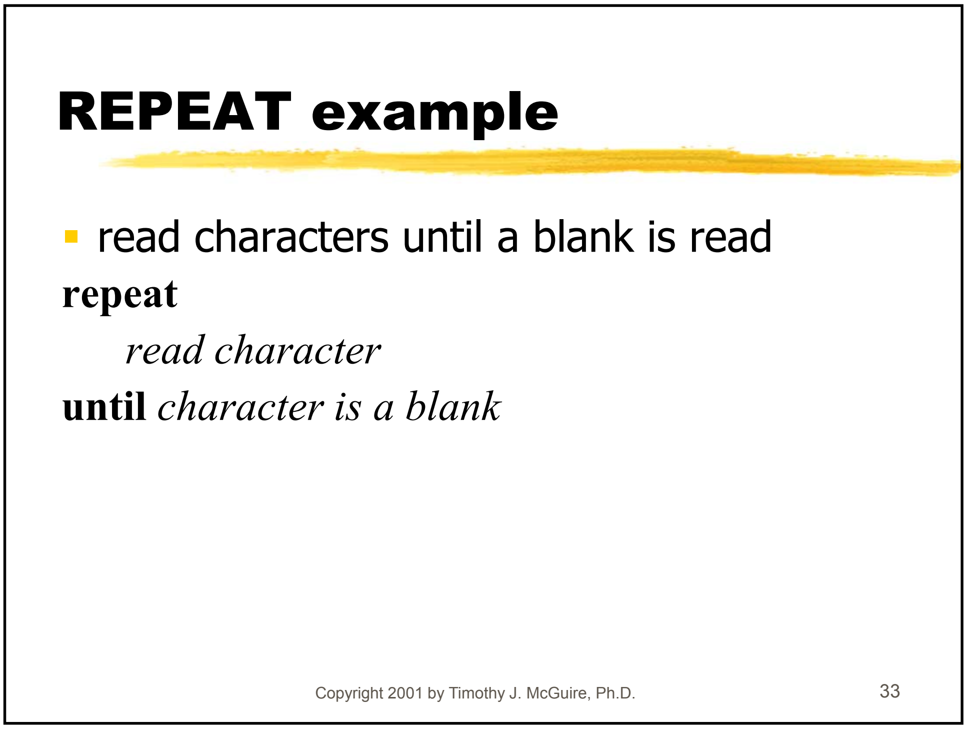
The REPEAT Loop

repeat

statements

until *condition*

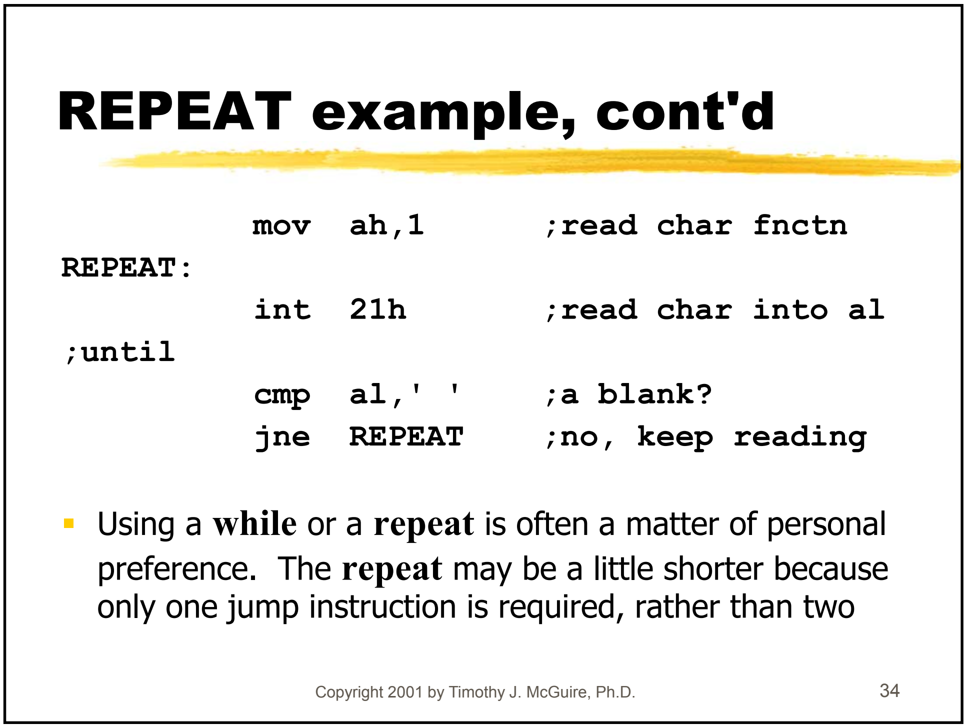
- The condition is checked at the bottom of the loop
- The loop executes until the condition is true
- The loop executes 1 or more times



REPEAT example

- read characters until a blank is read

```
repeat
    read character
until character is a blank
```



REPEAT example, cont'd

```
          mov  ah,1       ;read char fnctn
REPEAT:
          int  21h        ;read char into al
;until
          cmp  al,' '     ;a blank?
          jne  REPEAT     ;no, keep reading
```

- Using a **while** or a **repeat** is often a matter of personal preference. The **repeat** may be a little shorter because only one jump instruction is required, rather than two