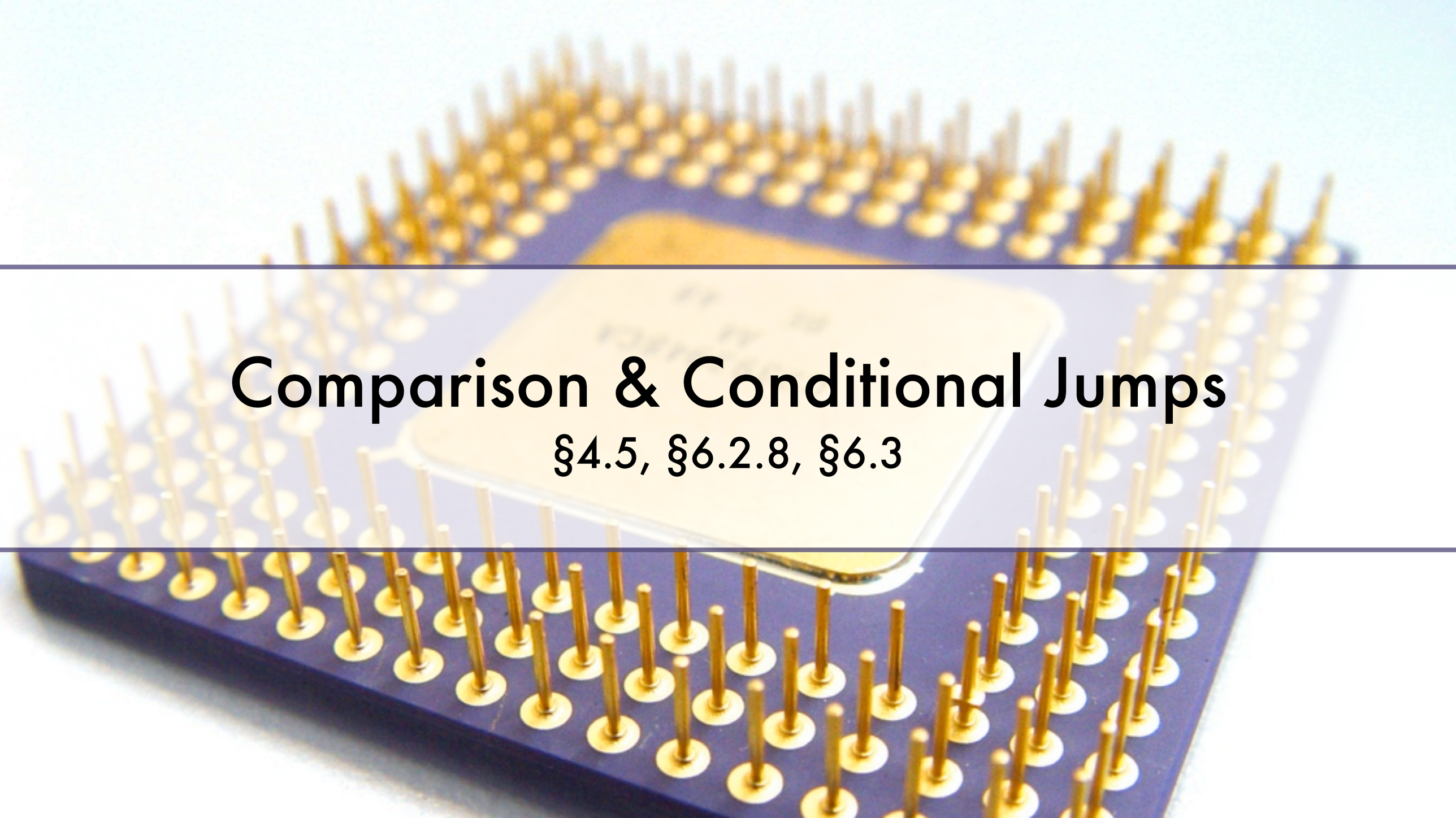
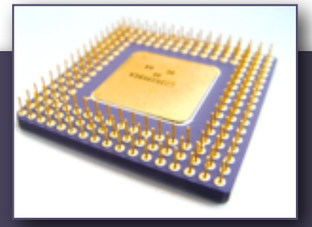


# Comparison & Conditional Jumps

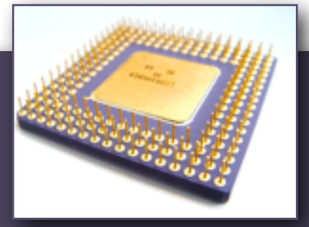
§4.5, §6.2.8, §6.3

# Conditional Jumps… For Real This Time



- You have seen one conditional jump instruction
  - How to implement *if* (ECX == 0) { ... } *else* { ... }
  - How to implement *while* (ECX == 0) { ... }
  - How to implement *do* { ... } *while* (ECX == 0)
  - And the same with the condition ECX ≠ 0
- **How do you perform comparisons other than "ECX == 0" and "ECX ≠ 0"?**
  - Perform a CMP (Compare) to set flags
  - Then perform a conditional jump

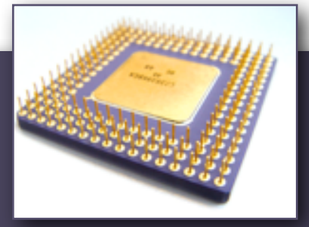
# Example 1: Comparison, Jumps



```
                ; Read a signed integer into EAX
                call ReadInt
                cmp eax, 100
                jge big
                ; If we reach here, eax < 100
                jmp done
big:            ; If we reach here, eax ≥ 100
done:           exit
```

JGE = Jump if Greater or Equal
(based on flags set by cmp)

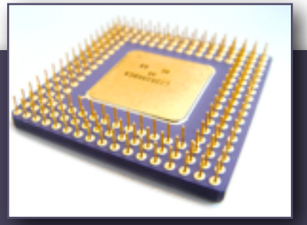
# Example 2: Comparison, Jumps



```
                ; Read a signed integer into EAX
                call ReadInt
                cmp eax, 10
                jle small
                ; If we reach here, eax > 10
                jmp done
small:          ; If we reach here, eax ≤ 10
done:           exit
```

JLE = Jump if Less or Equal
(based on flags set by cmp)

# Subtraction & Comparison

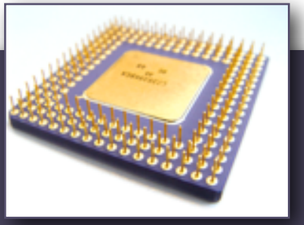


- How does the SUB instruction affect the flags?

| ***Unsigned*** | ZF | CF |
|---|---|---|
| Dest < Src | | |
| Dest = Src | | |
| Dest > Src | | |

| ***Signed*** | Flags |
|---|---|
| Dest < Src | SF ?=? OF |
| Dest = Src | ZF = ? |
| Dest > Src | SF ?=? OF |

# Subtraction & Comparison



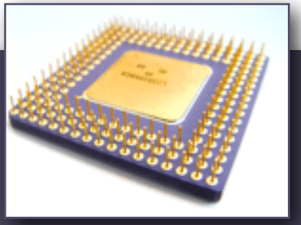
- How does the SUB instruction affect the flags?

| ***Unsigned*** | ZF | CF |
|---|---|---|
| Dest < Src | 0 | 1 |
| Dest = Src | 1 | 0 |
| Dest > Src | 0 | 0 |

| ***Signed*** | Flags |
|---|---|
| Dest < Src | SF ≠ OF |
| Dest = Src | ZF = 1 |
| Dest > Src | SF = OF |

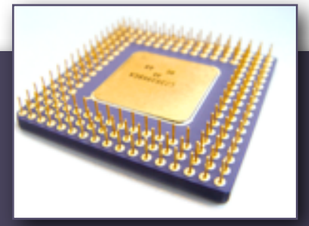
- Integer values can be compared by subtracting the values and then looking at the flags!
- The CMP (Compare) instruction subtracts values but does **not** store the result; it only sets flags

# Topics Covered in Notes:



- CMP instruction

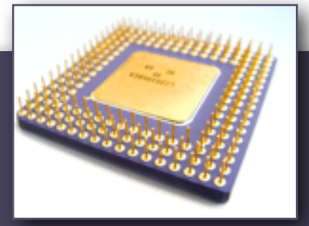
# Jumps Based on Specific Flags



| Mnemonic | Description | Flags |
|---|---|---|
| JZ | Jump if zero | ZF = 1 |
| JNZ | Jump if not zero | ZF = 0 |
| JC | Jump if carry | CF = 1 |
| JNC | Jump if not carry | CF = 0 |
| JO | Jump if overflow | OF = 1 |
| JNO | Jump if not overflow | OF = 0 |
| JS | Jump if signed | SF = 1 |
| JNS | Jump if not signed | SF = 0 |
| JP | Jump if parity (even) | PF = 1 |
| JNP | Jump if not parity (odd) | PF = 0 |

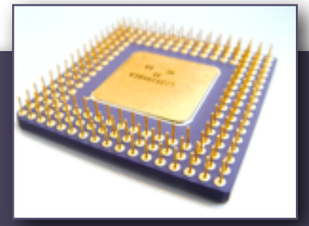
Activity 10 #1

# Jumps Based on Equality



| Mnemonic | Description |
|---|---|
| JE | Jump if equal (*leftOp* = *rightOp*) |
| JNE | Jump if not equal (*leftOp* ≠ *rightOp*) |
| JCXZ | Jump if CX = 0 |
| JECXZ | Jump if ECX = 0 |

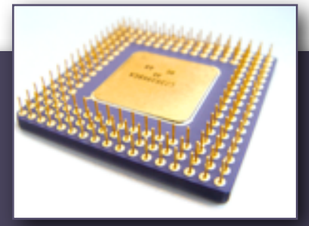
# Jumps Based on Signed Comparisons



| Mnemonic | Description |
|---|---|
| JG | Jump if greater (if *leftOp* > *rightOp*) |
| JNLE | Jump if not less than or equal (same as JG) |
| JGE | Jump if greater than or equal (if *leftOp* >= *rightOp*) |
| JNL | Jump if not less (same as JGE) |
| JL | Jump if less (if *leftOp* < *rightOp*) |
| JNGE | Jump if not greater than or equal (same as JL) |
| JLE | Jump if less than or equal (if *leftOp* <= *rightOp*) |
| JNG | Jump if not greater (same as JLE) |

Activity 10

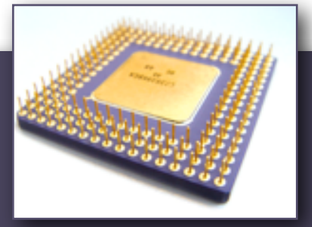
# Jumps Based on Unsigned Comparisons



| Mnemonic | Description |
|---|---|
| JA | Jump if above (if *leftOp* > *rightOp*) |
| JNBE | Jump if not below or equal (same as JA) |
| JAE | Jump if above or equal (if *leftOp* >= *rightOp*) |
| JNB | Jump if not below (same as JAE) |
| JB | Jump if below (if *leftOp* < *rightOp*) |
| JNAE | Jump if not above or equal (same as JB) |
| JBE | Jump if below or equal (if *leftOp* <= *rightOp*) |
| JNA | Jump if not above (same as JBE) |

Activity 10

# Jumps Based on Comparisons



## **Signed Comparisons** ("Greater/Less")

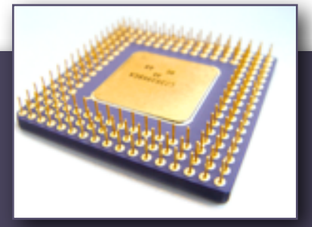
| Mnemonic | Description |
|---|---|
| JG | Jump if greater (if *leftOp* > *rightOp*) |
| JNLE | Jump if not less than or equal (same as JG) |
| JGE | Jump if greater than or equal (if *leftOp* >= *rightOp*) |
| JNL | Jump if not less (same as JGE) |
| JL | Jump if less (if *leftOp* < *rightOp*) |
| JNGE | Jump if not greater than or equal (same as JL) |
| JLE | Jump if less than or equal (if *leftOp* <= *rightOp*) |
| JNG | Jump if not greater (same as JLE) |

## **Unsigned Comparisons** ("Above/Below")

| Mnemonic | Description |
|---|---|
| JA | Jump if above (if *leftOp* > *rightOp*) |
| JNBE | Jump if not below or equal (same as JA) |
| JAE | Jump if above or equal (if *leftOp* >= *rightOp*) |
| JNB | Jump if not below (same as JAE) |
| JB | Jump if below (if *leftOp* < *rightOp*) |
| JNAE | Jump if not above or equal (same as JB) |
| JBE | Jump if below or equal (if *leftOp* <= *rightOp*) |
| JNA | Jump if not above (same as JBE) |

Activity 10 #2

# Conditional Jumps and Flags



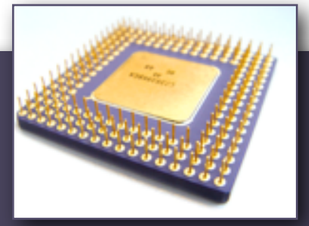
- Remember: JA, JB, JL, JG, etc. are based on **flags**
  - It's *conventional* to use `cmp` to set the flags
  - But if some other instruction changes the flags, the jump will be be based on its flags

```
.data
msg BYTE "3 < 5", 0
```

```
    mov ah, 3
    mov al, 5
    cmp ah, al
    jnl done

    mov edx, OFFSET msg
    call WriteString

done: exit
```

```
    mov ah, 3
    mov al, 5
    sub ah, al
    jnl done

    mov edx, OFFSET msg
    call WriteString

done: exit
```

# Translating Do-While Loops

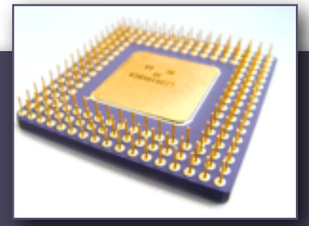


```
do {
    // body
} while (eax < ebx)
```

```
start:  ; body
        cmp eax, ebx
        jl start or jb start
```

*(signed)* *(unsigned)*

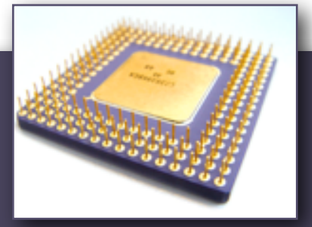
# Translating While Loops



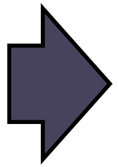
```
while (eax < ebx) {
    // body
}
```

```
start:  cmp eax, ebx
        jnl finish or jnb finish
        ; body
        jmp start
finish:
```

# Translating Counted Loops



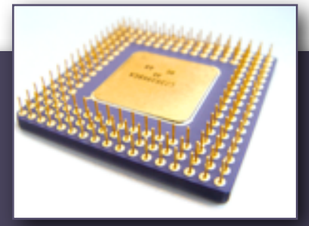
```
for (eax = 0; eax < 10; eax++) {
    // body
}
```



```
eax = 0
while (eax < 10) {
    // body
    eax++
}
```

```
        mov eax, 0
start:  cmp eax, 10
        jnl finish or jnb finish
        ; body
        inc eax
        jmp start
finish:
```

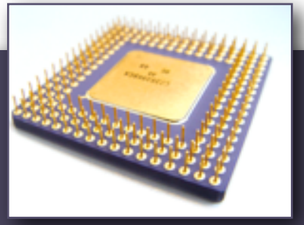
# Translating If Statements



```
if (eax < ebx) {
    // a
} else {
    // b
}
```

```
        cmp eax, ebx
        jnl zelse or jnb zelse
        ; a
        jmp zendif  ; Don't forget this!
zelse:  ; b
zendif:
```

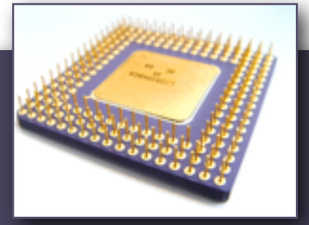
# Exercises



1. Will the jump be taken?

```
mov ah, 70h
add ah, 10h
jo some_label
```

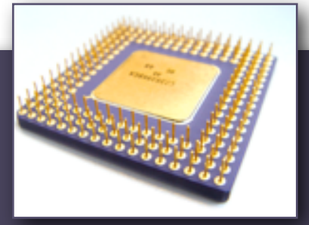
# Exercises



2. Will the jump be taken?

```
mov ah, -1
cmp ah, 5
jl some_label
```

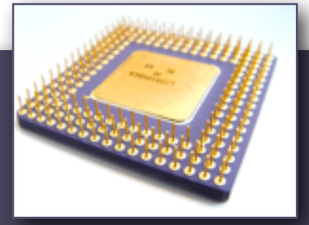
# Exercises



3. Will the jump be taken?

```
mov ah, -1
cmp ah, 5
jb some_label
```

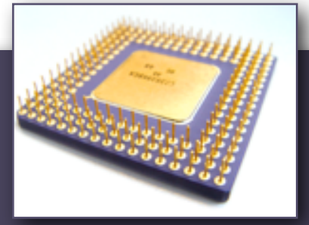
# Exercises



4. Will the jump be taken?

```
mov ah, 0FFh
cmp ah, -1
je some_label
```

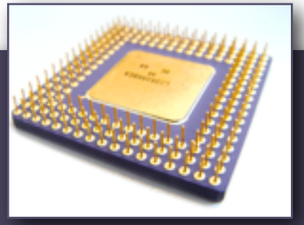
# Exercises



5. Will the jump be taken?

```
mov eax, 0FFh
cmp eax, -1
je some_label
```

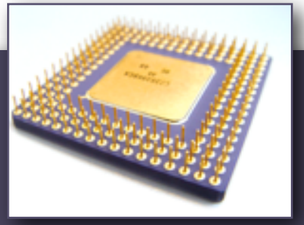
# Exercises



6. Will the jump be taken?

```
mov al, 100

mov ah, 25
add ah, 75

cmp ah, al
je some_label
```

# Exercises



7. Will the jump be taken?

```
mov al, 100
add al, 50
cmp al, 100
jg some_label
```