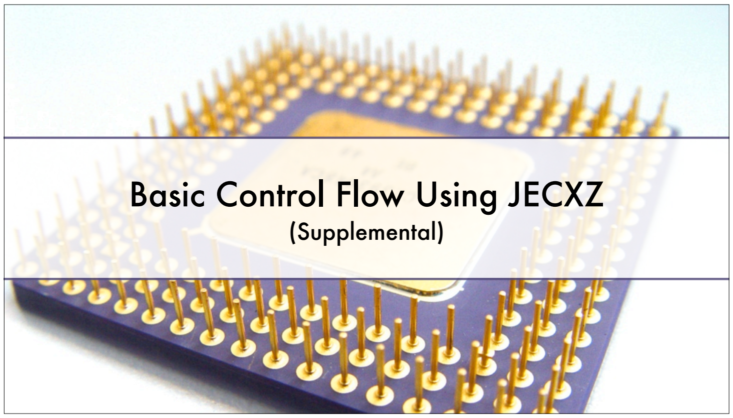
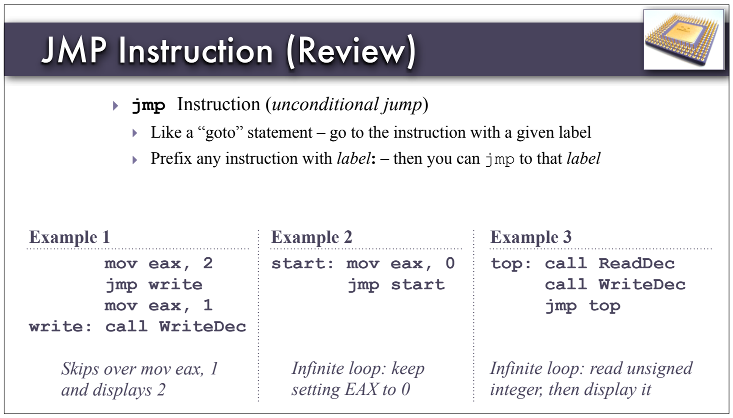


Basic Control Flow Using JECXZ

(Supplemental)



JMP Instruction (Review)

- **`jmp`** Instruction (*unconditional jump*)
  - Like a "goto" statement – go to the instruction with a given label
  - Prefix any instruction with *label***:** – then you can `jmp` to that *label*

Example 1

```
       mov eax, 2
       jmp write
       mov eax, 1
write: call WriteDec
```

Skips over mov eax, 1 and displays 2

Example 2

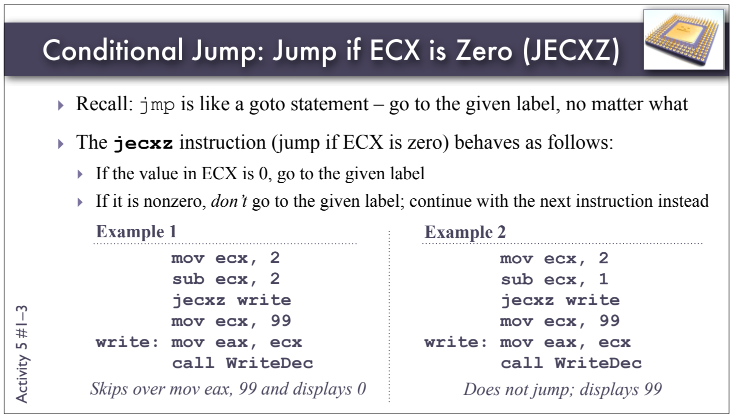
```
start: mov eax, 0
       jmp start
```

Infinite loop: keep setting EAX to 0

Example 3

```
top: call ReadDec
     call WriteDec
     jmp top
```

Infinite loop: read unsigned integer, then display it



Conditional Jump: Jump if ECX is Zero (JECXZ)

- Recall: `jmp` is like a goto statement – go to the given label, no matter what
- The **`jecxz`** instruction (jump if ECX is zero) behaves as follows:
  - If the value in ECX is 0, go to the given label
  - If it is nonzero, *don't* go to the given label; continue with the next instruction instead

Example 1

```
       mov ecx, 2
       sub ecx, 2
       jecxz write
       mov ecx, 99
write: mov eax, ecx
       call WriteDec
```

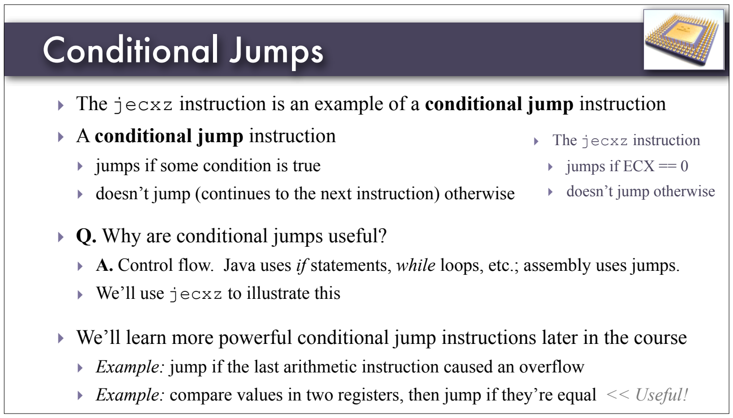
Skips over mov eax, 99 and displays 0

Example 2

```
       mov ecx, 2
       sub ecx, 1
       jecxz write
       mov ecx, 99
write: mov eax, ecx
       call WriteDec
```

Does not jump; displays 99

Activity 5 #1–3

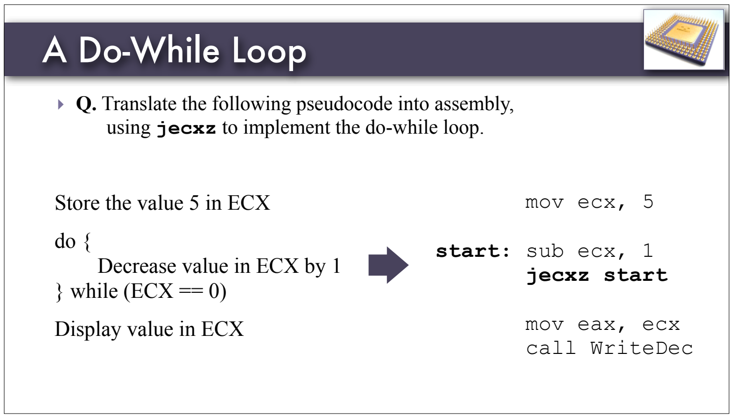


Conditional Jumps

- The `jecxz` instruction is an example of a **conditional jump** instruction
- A **conditional jump** instruction
  - jumps if some condition is true
  - doesn't jump (continues to the next instruction) otherwise

- The `jecxz` instruction
  - jumps if ECX == 0
  - doesn't jump otherwise

- **Q.** Why are conditional jumps useful?
  - **A.** Control flow. Java uses *if* statements, *while* loops, etc.; assembly uses jumps.
  - We'll use `jecxz` to illustrate this
- We'll learn more powerful conditional jump instructions later in the course
  - *Example:* jump if the last arithmetic instruction caused an overflow
  - *Example:* compare values in two registers, then jump if they're equal *<< Useful!*



A Do-While Loop

- **Q.** Translate the following pseudocode into assembly, using **`jecxz`** to implement the do-while loop.

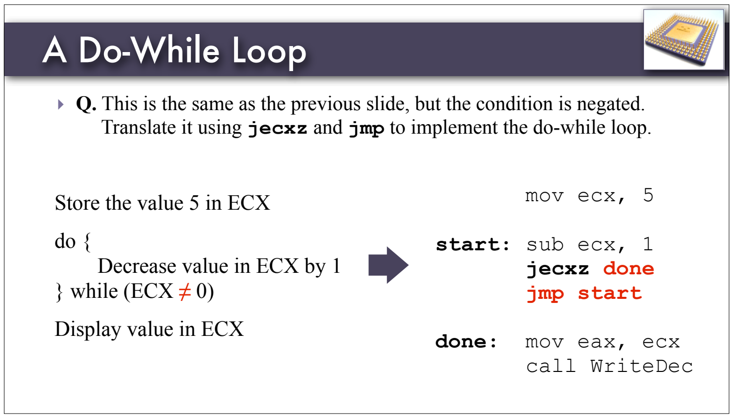
```
Store the value 5 in ECX
do {
    Decrease value in ECX by 1
} while (ECX == 0)
Display value in ECX
```

➡

```
        mov ecx, 5

start:  sub ecx, 1
        jecxz start

        mov eax, ecx
        call WriteDec
```



A Do-While Loop

- **Q.** This is the same as the previous slide, but the condition is negated. Translate it using **`jecxz`** and **`jmp`** to implement the do-while loop.

```
Store the value 5 in ECX
do {
    Decrease value in ECX by 1
} while (ECX ≠ 0)
Display value in ECX
```

➡

```
        mov ecx, 5

start:  sub ecx, 1
        jecxz done
        jmp start

done:   mov eax, ecx
        call WriteDec
```

Translating Do-While Loops

```
Do Thing A
do {
    Do Thing B
} while (ECX == 0)
Do Thing C

        ↓

        Do Thing A
label:  Do Thing B
        jecxz label
        Do Thing C
```

```
Do Thing A
do {
    Do Thing B
} while (ECX ≠ 0)
Do Thing C

        ↓

         Do Thing A
label1:  Do Thing B
         jecxz label2
         jmp label1
label2:  Do Thing C
```

Activity 5 #4

A While Loop

- Remember from Java:
  - do-while loops – test *after* executing the loop body
  - while loops – test *before* executing the loop body
- Example:

```
Store the value 5 in ECX                     mov ecx, 5

while (ECX ≠ 0) {                    start:  jecxz done
    Decrease value in ECX by 1   →           sub ecx, 1
}                                            jmp start

Display value in ECX                 done:   mov eax, ecx
                                             call WriteDec
```

A While Loop

- Remember from Java:
  - do-while loops – test *after* executing the loop body
  - while loops – test *before* executing the loop body
- Example:

```
Store the value 5 in ECX                     mov ecx, 5

while (ECX == 0) {                   start:  jecxz body
    Decrease value in ECX by 1   →           jmp done
}                                    body:   sub ecx, 1
                                             jmp start
Display value in ECX
                                     done:   mov eax, ecx
                                             call WriteDec
```

Translating While Loops

```
Do Thing A
while (ECX == 0) {
    Do Thing B
}
Do Thing C

        ↓

         Do Thing A
label1:  jecxz label2
         jmp label3
label2:  Do Thing B
         jmp label1
label3:  Do Thing C
```

```
Do Thing A
while (ECX ≠ 0) {
    Do Thing B
}
Do Thing C

        ↓

         Do Thing A
label1:  jecxz label3
         Do Thing B
         jmp label1
label3:  Do Thing C
```

Activity 5 #5

Summary: Translating Loops Involving ECX

```
Do Thing A
do {
    Do Thing B
} while (ECX == 0)
Do Thing C

    ↓

     Do Thing A
L1:  Do Thing B
     jecxz L1
     Do Thing C
```

```
Do Thing A
do {
    Do Thing B
} while (ECX ≠ 0)
Do Thing C

    ↓

     Do Thing A
L1:  Do Thing B
     jecxz L2
     jmp L1
L2:  Do Thing C
```

```
Do Thing A
while (ECX == 0) {
    Do Thing B
}
Do Thing C

    ↓

     Do Thing A
L1:  jecxz L2
     jmp L3
L2:  Do Thing B
     jmp L1
L3:  Do Thing C
```

```
Do Thing A
while (ECX ≠ 0) {
    Do Thing B
}
Do Thing C

    ↓

     Do Thing A
L1:  jecxz L3
     Do Thing B
     jmp L1
L3:  Do Thing C
```

An If Statement: General Form

- You can also implement an *if* statement using `jecxz`:

```
Do Thing A                              Do Thing A
if (ECX ≠ 0) {                          jecxz ???
    Do Thing B1             ifPart:     Do Thing B1
} else {               →                jmp ???
    Do Thing B2             elsePart:   Do Thing B2
}                           endPart:    Do Thing C
Do Thing C
```

An If Statement: General Form

```
Do Thing A
if (ECX ≠ 0) {
    Do Thing B1
} else {
    Do Thing B2
}
Do Thing C
```

⇨

```
           Do Thing A
           jecxz elsePart
ifPart:    Do Thing B1
           jmp endPart
elsePart:  Do Thing B2
endPart:   Do Thing C
```

These are just ordinary labels.
You don't have to call them ifPart, elsePart, etc.
Any label—L1, or dog, or foo—will work (but it's less readable).

Translating If Statements

```
Do Thing A
if (ECX ≠ 0) {
    Do Thing B1
} else {
    Do Thing B2
}
Do Thing C
```

⇨

```
           Do Thing A
           jecxz elsePart
ifPart:    Do Thing B1
           jmp endPart
elsePart:  Do Thing B2
endPart:   Do Thing C
```

```
Do Thing A
if (ECX == 0) {
    Do Thing B1
} else {
    Do Thing B2
}
Do Thing C
```

⇨

```
           Do Thing A
           jecxz ifPart
           jmp elsePart
ifPart:    Do Thing B1
           jmp endPart
elsePart:  Do Thing B2
endPart:   Do Thing C
```

Activity 5 #6

Administrivia

- **Homework 1** was due at 2:00 – late submission cutoff is 2 p.m. Sunday
- Meet in the **Lab on Monday** (2119 and 2122 Shelby)
  - Go to either one – wherever you can find a seat
  - If you want to work on your laptop, bring it