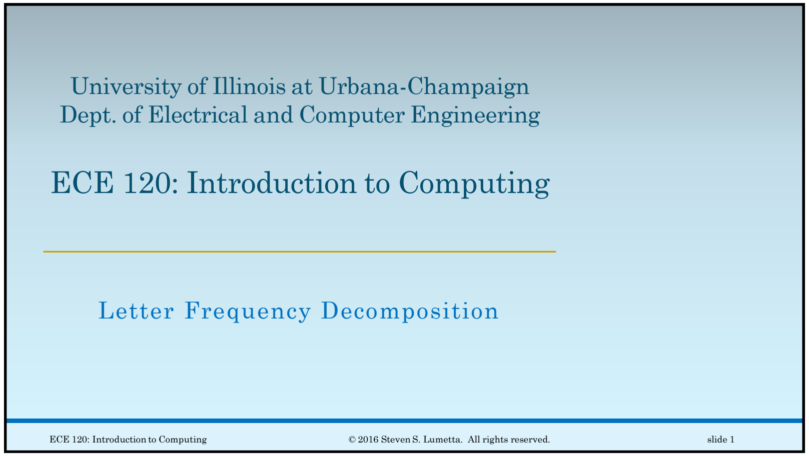
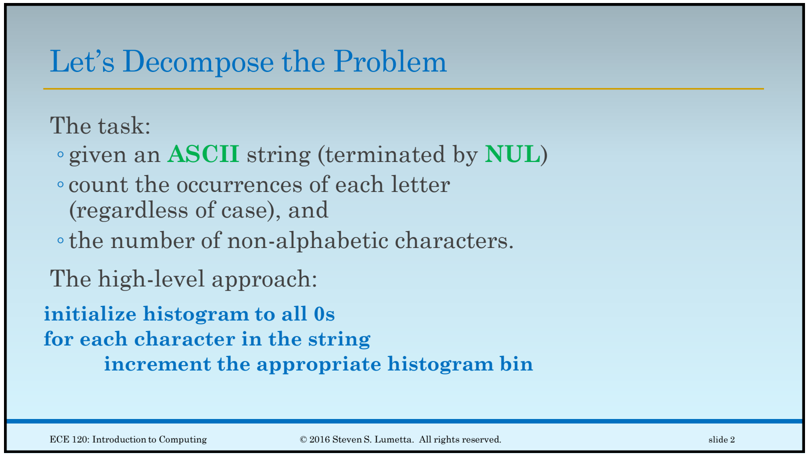


University of Illinois at Urbana-Champaign
Dept. of Electrical and Computer Engineering

ECE 120: Introduction to Computing

Letter Frequency Decomposition



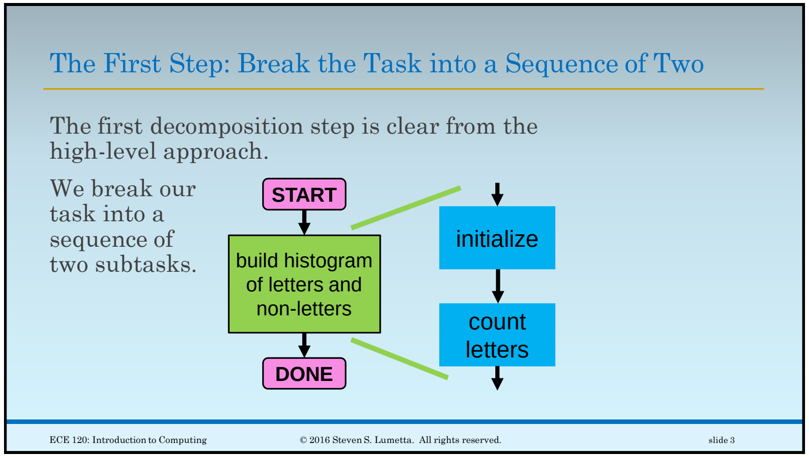
Let's Decompose the Problem

The task:

- given an **ASCII** string (terminated by **NUL**)
- count the occurrences of each letter (regardless of case), and
- the number of non-alphabetic characters.

The high-level approach:

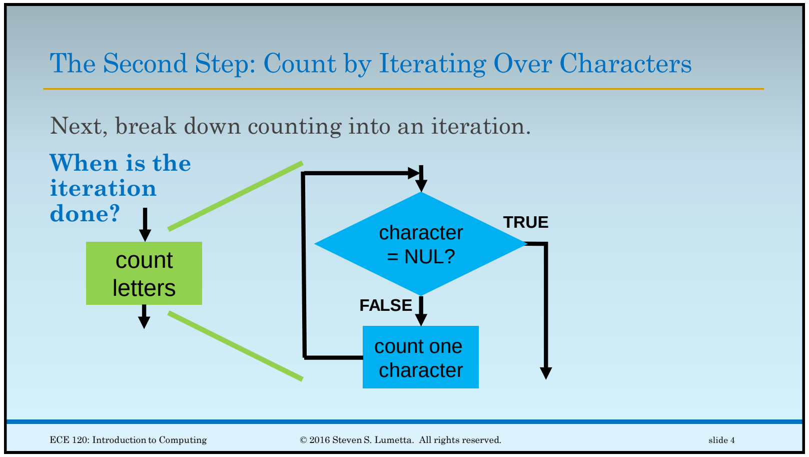
```
initialize histogram to all 0s
for each character in the string
        increment the appropriate histogram bin
```



The First Step: Break the Task into a Sequence of Two

The first decomposition step is clear from the high-level approach.

We break our task into a sequence of two subtasks.



The Second Step: Count by Iterating Over Characters

Next, break down counting into an iteration.

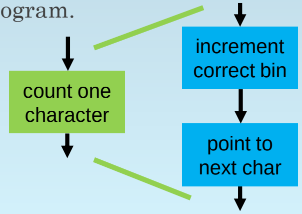
When is the iteration done?

Break Down Counting a Character into Two Steps

Counting one character involves two steps.

First, we must **increment one bin** in the histogram.

Then we must **advance our pointer** to the next character in the string.



How to Choose a Bin: Use a Conditional Construct

How can we determine which histogram bin to increment?

The answer depends on the character.

We **need to use conditional constructs**.

But how?

Let's take a look at the **ASCII** sequence.

The ASCII Table Breaks into Five Regions

Here is an abbreviated version of the **ASCII** table.

x00		x40	x41		x5A	x5B		x60	x61		x7A	x7B		x7F
NUL	…	@	A	…	Z	[	…	`	a	…	z	{	…	DEL

The characters divide into five groups.

The **blue** groups are letters.

The **green** groups are non-alphabetic.

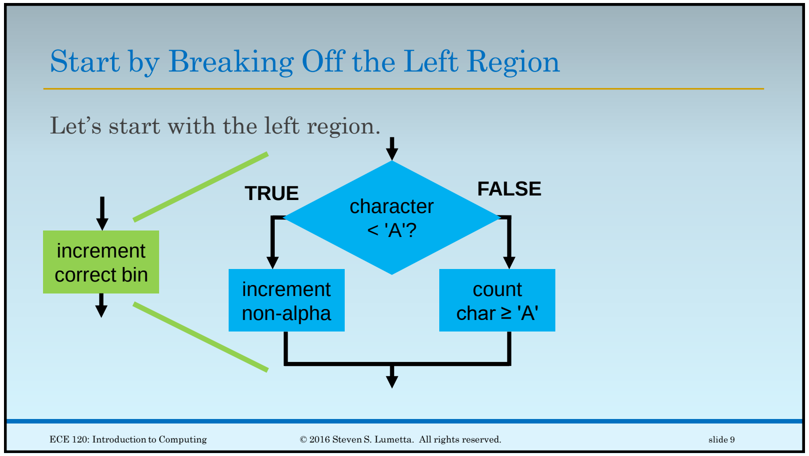
We can **use the vertical lines as conditions**.

Start by Breaking Off the Left Region

x00		x40	x41		x5A	x5B		x60	x61		x7A	x7B		x7F
NUL	…	@	A	…	Z	[	…	`	a	…	z	{	…	DEL

Notice that
- if a character is less than 'A',
- the character is not a letter.

Let's **start with the leftmost region**.



Start by Breaking Off the Left Region

Let's start with the left region.

increment correct bin

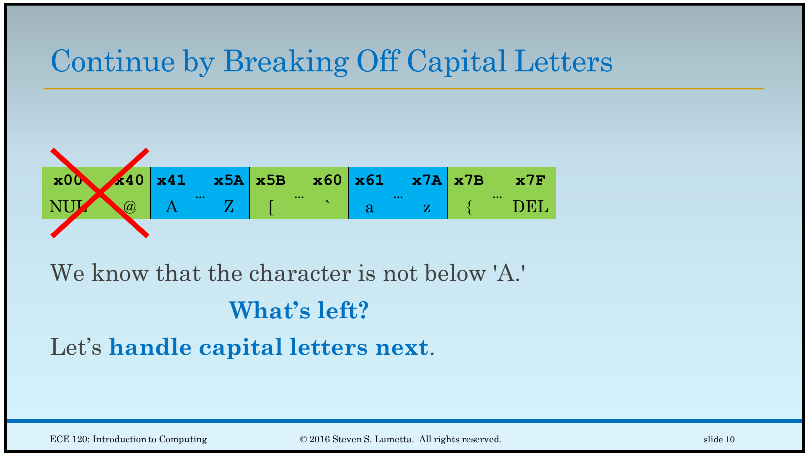
TRUE

character < 'A'?

FALSE

increment non-alpha

count char ≥ 'A'



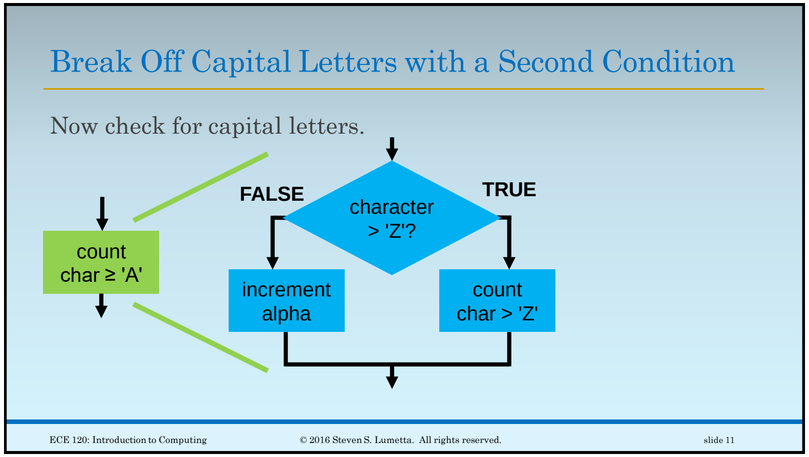
Continue by Breaking Off Capital Letters

~~x00 x40~~	x41 x5A	x5B x60	x61 x7A	x7B x7F
~~NUL … @~~	A … Z	[… `	a … z	{ … DEL

We know that the character is not below 'A.'

What's left?

Let's **handle capital letters next**.



Break Off Capital Letters with a Second Condition

Now check for capital letters.

count char ≥ 'A'

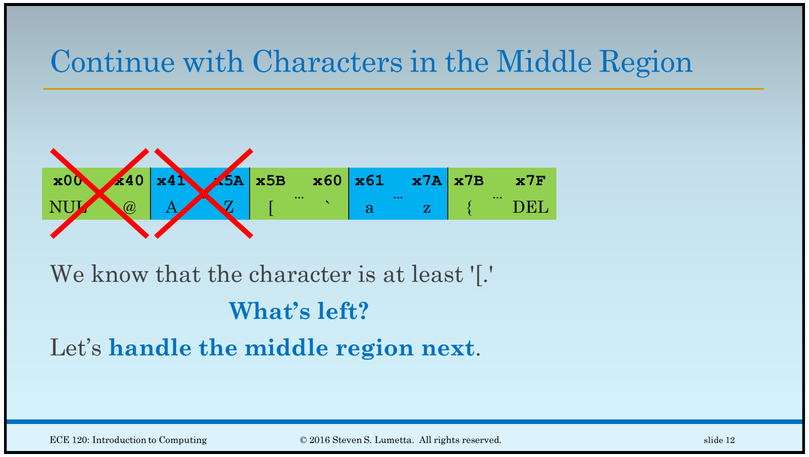
FALSE

character > 'Z'?

TRUE

increment alpha

count char > 'Z'



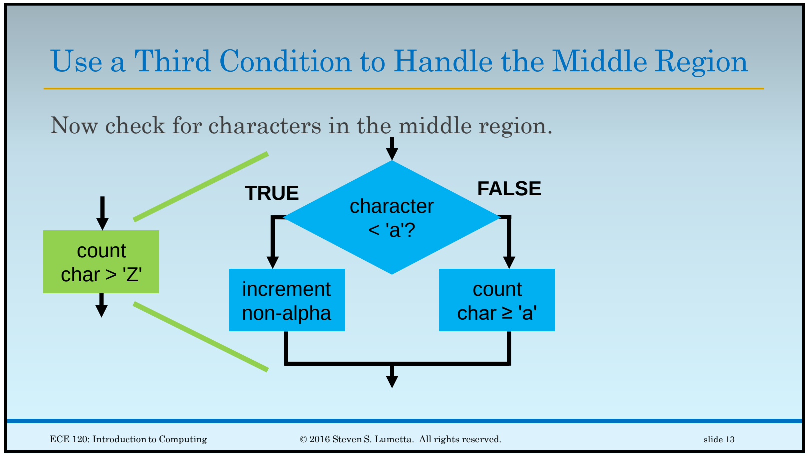
Continue with Characters in the Middle Region

~~x00 x40~~	~~x41 x5A~~	x5B x60	x61 x7A	x7B x7F
~~NUL … @~~	~~A … Z~~	[… `	a … z	{ … DEL

We know that the character is at least '[.'

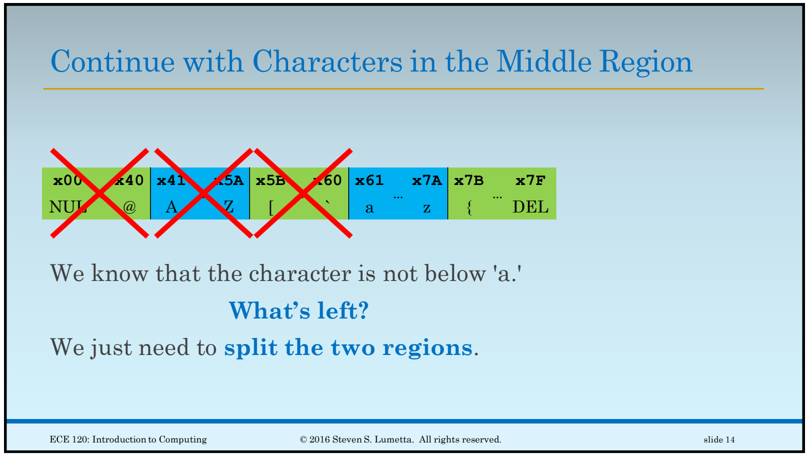
What's left?

Let's **handle the middle region next**.



Use a Third Condition to Handle the Middle Region

Now check for characters in the middle region.



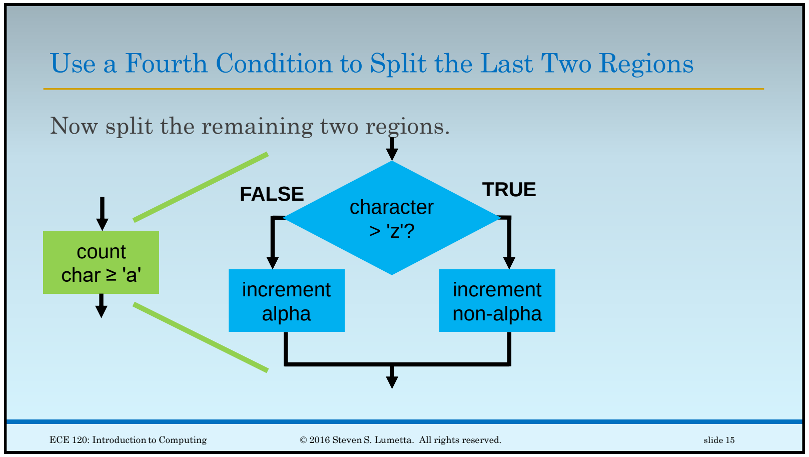
Continue with Characters in the Middle Region

~~x00 … x40~~	~~x41 … x5A~~	~~x5B … x60~~	x61 … x7A	x7B … x7F
~~NUL … @~~	~~A … Z~~	~~[… `~~	a … z	{ … DEL

We know that the character is not below 'a.'

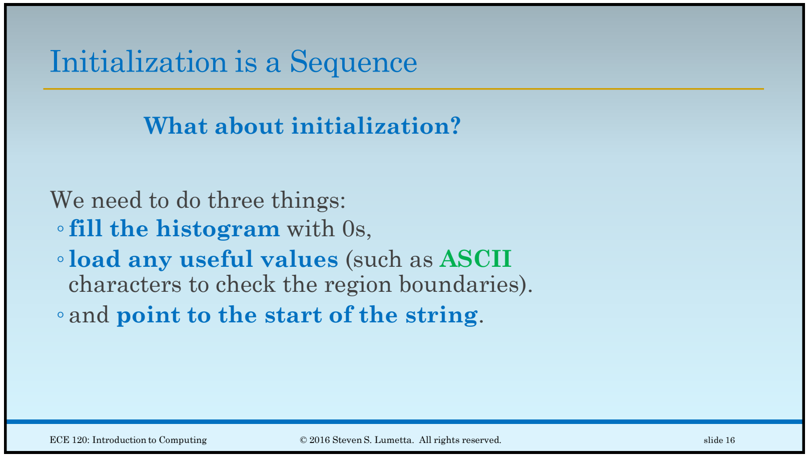
What's left?

We just need to **split the two regions**.



Use a Fourth Condition to Split the Last Two Regions

Now split the remaining two regions.



Initialization is a Sequence

What about initialization?

We need to do three things:

- **fill the histogram** with 0s,
- **load any useful values** (such as **ASCII** characters to check the region boundaries).
- and **point to the start of the string**.

Filling the Histogram: a Sequence and an Iteration

How do we fill the histogram?

We have 27 bins (26 letters + 1 non-alpha).

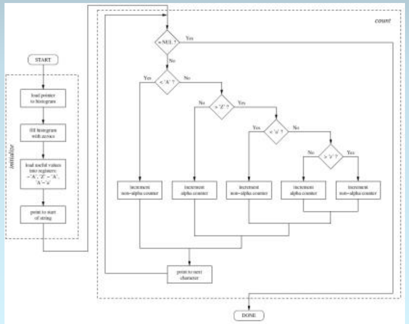
We should use an **iteration**.

But again, we need a pointer to the histogram.

So:

- **point a register to the histogram,**
- **then iterate over all bins.**

The Final Flow Chart


START
initialize
load pointer to histogram
fill histogram with zeroes
load useful values into registers: −'A', 'Z' − 'A', 'A' − 'a'
point to start of string
count
= NUL ?
Yes
No
< 'A' ?
Yes
No
> 'Z' ?
No
Yes
< 'a' ?
Yes
No
> 'z' ?
No
Yes
increment non−alpha counter
increment alpha counter
increment non−alpha counter
increment alpha counter
increment non−alpha counter
point to next character
DONE