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?

- character = NUL?
  - TRUE
  - character ≠ NUL?
    - count one character
    - FALSE
    - count letters
      - done

- START
- initialize
- build histogram of letters and non-letters
- count letters
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.
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

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.

```
TRUE

increment

character < 'A'?

FALSE

increment

non-alpha

count

char ≥ 'A'
```

Increment correct bin.

Continue by Breaking Off Capital Letters

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

What's left?

Let's handle capital letters next.

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

Continue with Characters in the Middle Region

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

What's left?

Let's handle the middle region next.

```
x00 x40 x41 x5A x5B x60 x61 x7A x7B x7F
NUL A Z [ ` a z { DEL
```
Use a Third Condition to Handle the Middle Region

Now check for characters in the middle region.

```
count char > 'Z'

TRUE

char < 'a'? count char ≥ 'a'

increment non-alpha increment alpha
```

Continue with Characters in the Middle Region

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.

```
count char ≥ 'a'

FALSE

char > 'z'? increment alpha increment non-alpha
```

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.