Review the Problem to Be Solved

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

Where Are the Pieces in Memory?

Let’s start with some notes about where we want to store information

<table>
<thead>
<tr>
<th>STRING</th>
<th>the start of the string</th>
</tr>
</thead>
<tbody>
<tr>
<td>x3000</td>
<td>the start of our code</td>
</tr>
<tr>
<td>HIST</td>
<td>non-alpha histogram bin</td>
</tr>
<tr>
<td>HIST + 1 to 26</td>
<td>alpha bins A to Z (in order)</td>
</tr>
</tbody>
</table>

What Shall We Keep in the Registers?

For the counting part, we will use registers as follows

- R0: histogram pointer (HIST)
- R1: string pointer (moves)
- R2: current character from string
- R3, R4, R5: ASCII constants (to be chosen)
- R6: temporary
Get a Pointer to the Histogram into R0

Slide 5

We need to initialize R0 to HIST.

Is there an LC-3 instruction for that?

We Also Need to Fill the Histogram with 0s

Slide 6

The next step: fill the histogram with 0s.

We need registers.

Let's reuse a few (so far, only R0 is initialized).

- R1: a loop counter (27 iterations)
- R2: current histogram bin to fill
- R6: the number 0 (to store)

Prepare Our Registers to Initialize the Histogram

Slide 7

To set R6 to 0, use an AND.

Prepare Our Registers to Initialize the Histogram

Slide 8

Now, we need to initialize R6 to 0, R1 to #27, and R2 to HIST.

What about R1?

Let's just store #27 somewhere and use an LD.

NUM_BINS.FILL #27 (just before .END)
Prepare Our Registers to Initialize the Histogram

- ORIG x3000
- LEA R0,HIST
- AND R6,R6,#0
- LD R1,NUM_BINS
- ADD R2,R0,#0

Now, we need to initialize R6 to 0, R1 to #27, and R2 to HIST.

And what about R2?

Remember that R0 already has the value HIST!

We’re Ready to Fill the Histogram with 0s

Remember our register contents:
- R1 a loop counter (27 iterations)
- R2 current histogram bin to fill
- R6 the number 0 (to store)

In our loop body, we write one 0 (from R6) to a bin at the memory location pointed to by R2.
Then we point to the next bin (increment R2). Then we decrement our loop counter (R1). Finally, we loop until the counter reaches 0.

Fill One Histogram Bin with 0

- ORIG x3000
- LEA R0,HIST
- AND R6,R6,#0
- LD R1,NUM_BINS
- ADD R2,R0,#0
- HFLOOP ; (hist fill loop)
- STR R6,R2,#0

Point to the Next Histogram Bin

- ORIG x3000
- LEA R0,HIST
- AND R6,R6,#0
- LD R1,NUM_BINS
- ADD R2,R0,#0
- HFLOOP ; (hist fill loop)
- STR R6,R2,#0
- ADD R2,R2,#1

Is there an LC-3 instruction for that?
Decrement the Loop Counter

Decrement the loop counter.

Is there an LC-3 instruction for that?

Branch Backward Until We Finish Filling the Histogram

Branch backward until we have written 27 bins.

Is there an LC-3 instruction for that?

R1 started at #27.

We Still Have Initialization Work to Do

What about these other registers?

R1  string pointer (moves)
R2  current character from string
R3, R4, R5  ASCII constants (to be chosen)
R6  temporary

Let's initialize them now.
(No need to initialize R2 nor R6.)

Initialize the Remaining Registers with LD

Initialize the other registers using LD.

(No need to initialize R2 nor R6.)

Note use of label STRING as a .FILL value.
Ready to Count Letters?

Now we are finally ready to count letters!

Before We Can Count, We Must Load a Character

The first step?
- Load a character from the string, and
- check if it’s NUL.

Load a Character from the String

COUNTLOOP
LDR R2,R1,#0

Remember that R1 points to the next character in the string.

Also remember that we want the character in R2.

If We Find a NUL, We are Done

COUNTLOOP
LDR R2,R1,#0
BRz DONE
Now We Can Classify the Character

We need to compare with capital A. Let's define $R_3$ as \texttt{-'@'} ...

We store the difference (original character minus \texttt{'@'}) back in $R_2$, so A through Z become 1 through 26.

Subtract $\texttt{@}$ to Compare with Capital A

Remember the ASCII table?

Subtracting \texttt{'@'} allows us to check for non-alphabetic characters in the left region.

Branch Unless We Have a Character in the Left Region

COUNTLOOP
LDR R2,R1,#0
BRz DONE
ADD R2,R2,R3

Branch forward if the character is not in the left non-alphabetic region.

What is the branch condition?
Time to Increment the Non-Alpha Histogram Bin

If the result is not positive,
- the character is in the left region and
- is not a letter.

So we can increment the non-alpha bin (at HIST).

Increment Memory Location x3100 (Non-Alpha Bin)

COUNTLOOP
LDR R2,R1,#0
BRz DONE
ADD R2,R2,R3
BRp AT_LEAST_A
NON_ALPHA
LDR R6,R0,#0
ADD R6,R6,#1

Increment Memory Location x3100 (Non-Alpha Bin)

COUNTLOOP
LDR R2,R1,#0
BRz DONE
ADD R2,R2,R3
BRp AT_LEAST_A
NON_ALPHA
LDR R6,R0,#0
ADD R6,R6,#1
STR R6,R0,#0

Increment Memory Location x3100 (Non-Alpha Bin)

COUNTLOOP
LDR R2,R1,#0
BRz DONE
ADD R2,R2,R3
BRp AT_LEAST_A
NON_ALPHA
LDR R6,R0,#0
ADD R6,R6,#1
STR R6,R0,#0
We Are Done with That Character

We are done counting that character. The loop is inside the first task shown here (the one labeled "increment correct bin"). So now we need to **point to the next character**... 

We Need to Check for a Capital Letter

Next, we compare with capital Z.

Subtract Z to Make the Next Comparison

This time, we want to subtract 'Z'. But we already subtracted '@', so now we add 'a' - 'Z' (let's keep this value in R4). We discard the result (store the result in R6).
Add (@ – Z) to Compare with Capital Z

AT_LEAST_A
ADD R6,R2,R4

Add R4 ('@' – 'Z') to R2 and write the sum into R6.

Compare with capital Z.

Time to Increment the One Letter's Histogram Bin

If the result is not positive, the character is a capital letter.

What bin should we increment?
(Hint: R2 now holds 1 to 26 for A to Z.)

The bin at address HIST + R2.

Branch Unless We Have a Capital Letter

AT_LEAST_A
ADD R6,R2,R4
BRp MORE_THAN_Z

What is the branch condition?

Branch forward if the character is not a capital letter.

Remember: we just calculated (original character – 'Z')

Increment One Letter's Histogram Bin

AT_LEAST_A
ADD R6,R2,R4
BRp MORE_THAN_Z
ALPHA
ADD R2,R2,R0

Where can we put the bin pointer?

We only need R2 to find the right bin.

Increment memory at HIST + R2 (R0 + R2).

First, we need to calculate a bin pointer.
Increment One Letter’s Histogram Bin

**AT_LEAST_A**
ADD R6,R2,R4
BRp MORE_THAN_Z
**ALPHA**
ADD R2,R2,R0
LDR R6,R2,#0

Same answer as last time: load, modify, store.

Increment memory at address pointed to by R2.

Is there an LC-3 instruction for that?

ADD R6,R2,R4
BRp MORE_THAN_Z
**ALPHA**
ADD R2,R2,R0
LDR R6,R2,#0
ADD R6,R6,#1

Slide 37

Increment memory at address pointed to by R2.

And now increment the value.

Slide 38

Increment One Letter’s Histogram Bin

**AT_LEAST_A**
ADD R6,R2,R4
BRp MORE_THAN_Z
**ALPHA**
ADD R2,R2,R0
LDR R6,R2,#0
ADD R6,R6,#1
STR R6,R2,#0

Increment memory at address pointed to by R2.

And put the new value back.

Slide 39

We Are Done with That Character

As before, we are done with that character. So now we need to **point to the next character**...
Subtract x60 to Make the Next Comparison

We want to subtract \( x_{60} \) (backquote, `'`).
But we already subtracted `'@'` from \( R_2 \), so now add `'@' - `'` (let’s keep this value in \( R_5 \)).
Let’s write the result back to \( R_2 \) so that lower case letters produce values 1 to 26 in \( R_2 \).

Add (\(@ - `\)) to Compare with Lower Case a

\[
\text{MORE_THAN_Z} \\
\text{ADD } R_2, R_2, R_5
\]

Add \( R_5 ('@' - `') \) to \( R_2 \) and write the sum back to \( R_2 \).
When Do We Have a Character in the Middle Region?

We just wrote (original character minus x60) into R2.

Under what conditions (N, Z, P) do we have a character in the middle region?

N and Z

Branch If We Have a Character in the Middle Region

MORE_THAN_Z
ADD R2,R2,R5
BRnz NON_ALPHA

So what is the branch condition?

Handle characters in the middle region.

Remember the label that we created earlier (for incrementing the non-alpha bin)?

How Can We Increment the Non-Alpha Bin?

So for conditions N or Z, we want to increment the non-alpha bin.

How?

Didn’t we already write that code?

Let’s just branch to it!

We Need to Check for a Lower Case Letter

Next, we compare with lower case z.

FALSE
increment alpha

TRUE
increment non-alpha

character > ‘z’?
Subtract z to Make the Next Comparison

This time, we want to subtract ‘z’.
But we already subtracted ‘1’, so now we add ‘1’ – ‘z’ (it’s already in R4).
We discard the result (store the result in R6).

Add ( – z) to Compare with Lower Case z

MORE_THAN_Z
ADD R2,R2,R5
BRnZ NON_ALPHA
ADD R6,R2,R4

Compare with lower case z.

When Do We Have a Lower Case Letter?

We just wrote (original character minus ‘z’) into R6.
Under what conditions (N, Z, P) do we have a lower case letter?
N and Z

How Can We Increment the Right Letter's Bin?

So for conditions N or Z, we want to increment one of the letter's histogram bins.
How?
Didn’t we already write that code?
Let’s just branch to it!
How Can We Increment the Right Letter’s Bin?

Let’s be clear:
• We are able to reuse the code because we designed the code to be reusable.
• In both cases, R0 points to the histogram, and R2 is 1 to 26 for the letter.

We Know that the Character is Not a Letter

At this point, we know that the original character was not a letter.

So … ?
Branch (unconditionally) to the code that increments the non-alpha histogram bin.

Branch If We Have a Lower Case Letter

What is the branch condition?

Handle lower case letters.

We created a second label for incrementing a letter’s bin.

Branch to the Code for Non-Alphabetic Characters

Handle the last region.

Again, just use the label created earlier.
Next, Advance the String Pointer

We are now finished with the upper task. We can write the code to point to the next character.

Our Loop Body is Complete

And now we’re done with counting a character and advancing the string pointer, so we can return to the start of our loop.

Advance the String Pointer to the Next Character

GET_NEXT
ADD R1,R1,#1

Advance the string pointer (in R1).

Is there an LC-3 instruction for that?

Return to the Start of the Loop

GET_NEXT
ADD R1,R1,#1
BRnzp COUNTLOOP

Return to the start of the loop.

Is there an LC-3 instruction for that?
We Need a HALT and Some Data

GET_NEXT
ADD R1,R1,#1
BRnzp COUNTLOOP
DONE
HALT
; earlier .FILLs here
HIST .BLKW #27
STRING .STRINGZ "hi"

We need a HALT and some data.
The full program is available online.

Our Work Here is Done!

That's it!

Unless you want to convert it to binary by hand, of course...