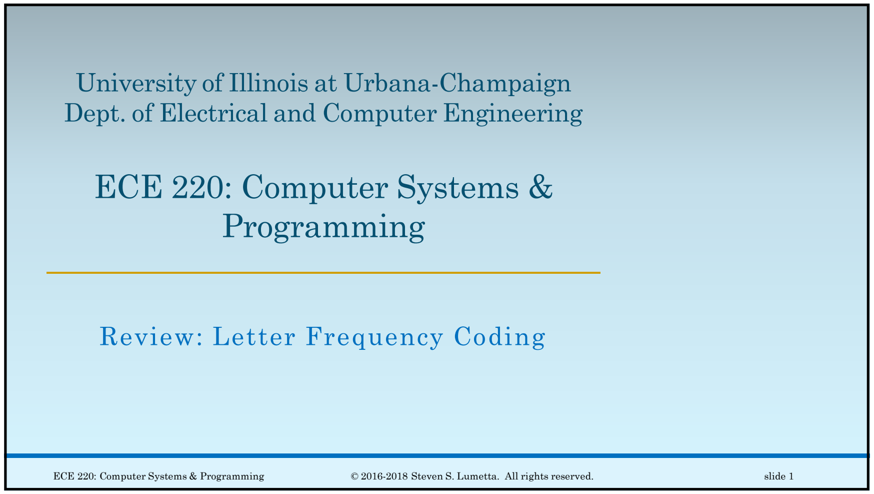
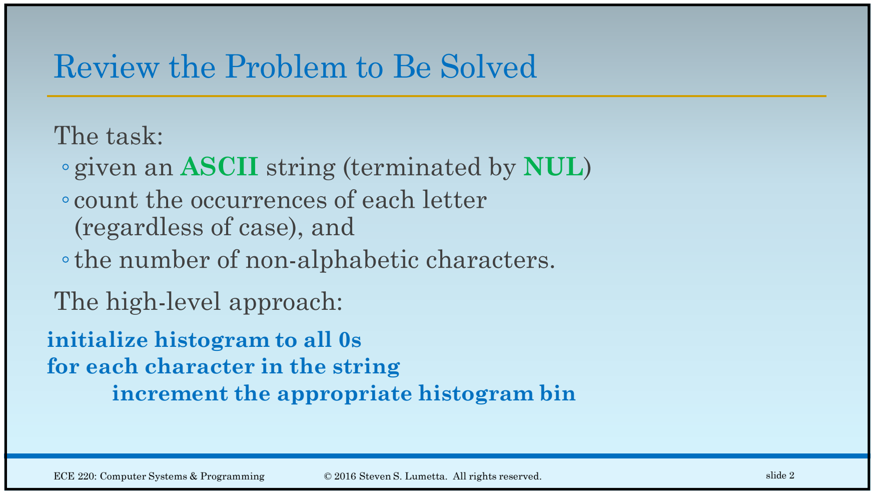


University of Illinois at Urbana-Champaign
Dept. of Electrical and Computer Engineering

ECE 220: Computer Systems & Programming

Review: Letter Frequency Coding



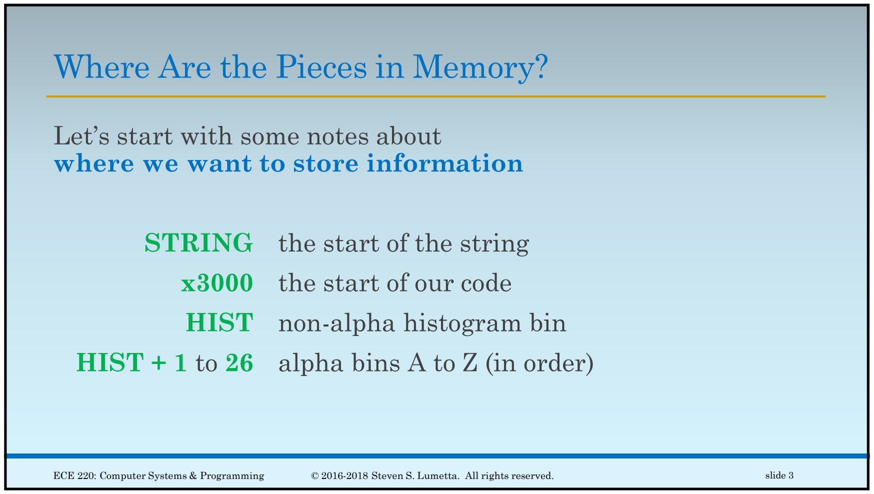
Review the Problem to Be Solved

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:

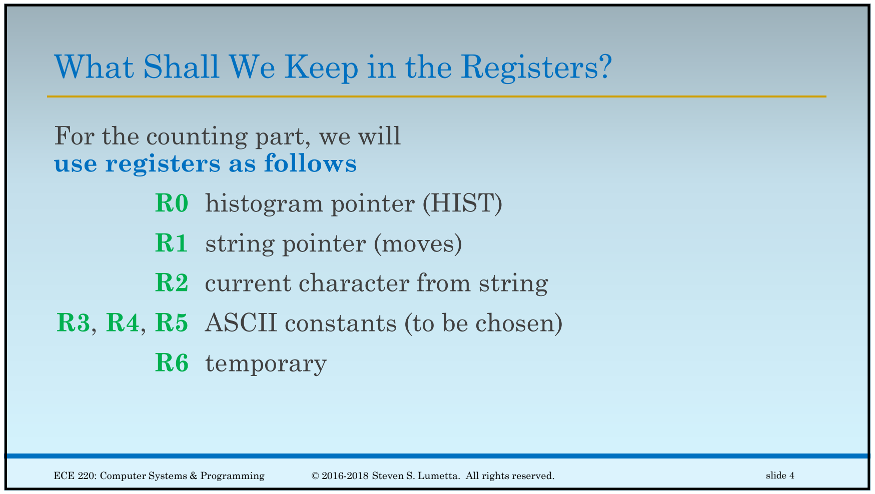
```
initialize histogram to all 0s
for each character in the string
        increment the appropriate histogram bin
```



Where Are the Pieces in Memory?

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

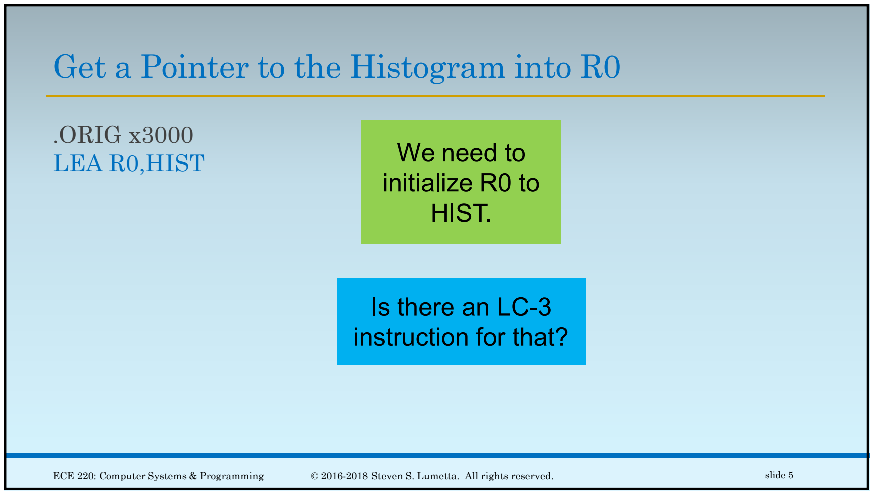
STRING	the start of the string
x3000	the start of our code
HIST	non-alpha histogram bin
HIST + 1 to **26**	alpha bins A to Z (in order)



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

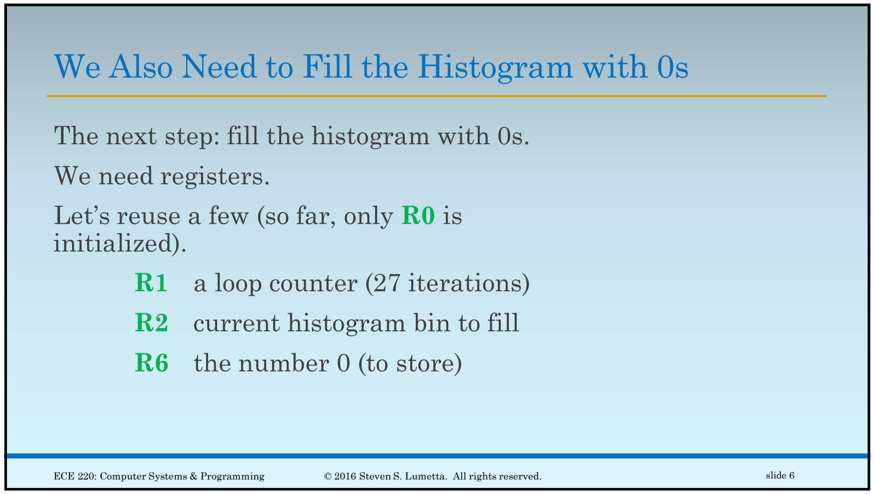


Get a Pointer to the Histogram into R0

```
.ORIG x3000
LEA R0,HIST
```

We need to initialize R0 to HIST.

Is there an LC-3 instruction for that?



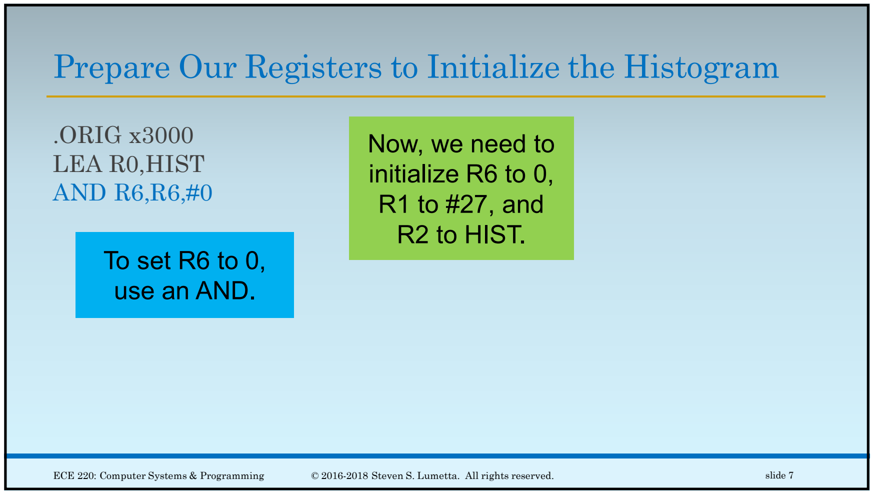
We Also Need to Fill the Histogram with 0s

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)

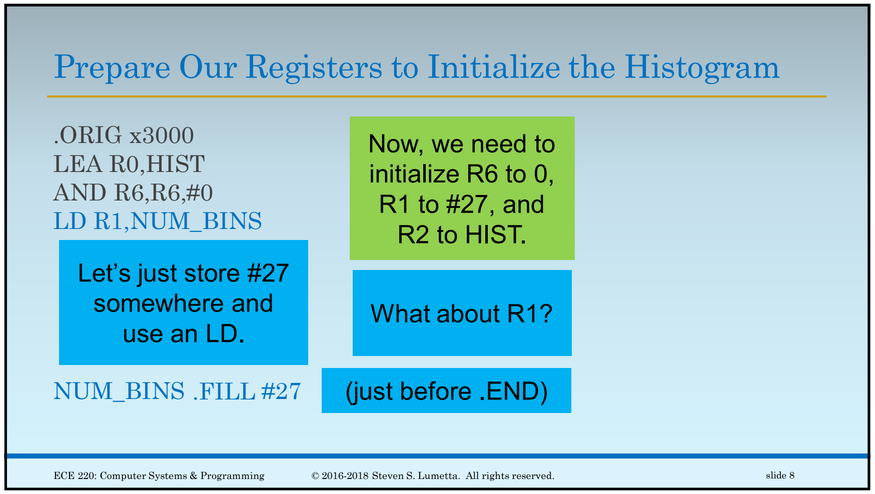


Prepare Our Registers to Initialize the Histogram

```
.ORIG x3000
LEA R0,HIST
AND R6,R6,#0
```

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

To set R6 to 0, use an AND.



Prepare Our Registers to Initialize the Histogram

```
.ORIG x3000
LEA R0,HIST
AND R6,R6,#0
LD R1,NUM_BINS
```

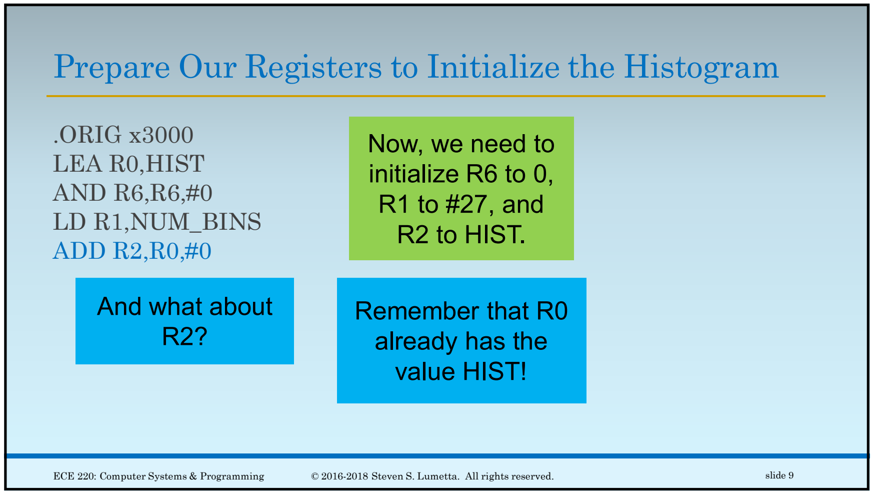
Now, we need to initialize R6 to 0, R1 to #27, and R2 to HIST.

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

What about R1?

```
NUM_BINS .FILL #27
```

(just before .END)



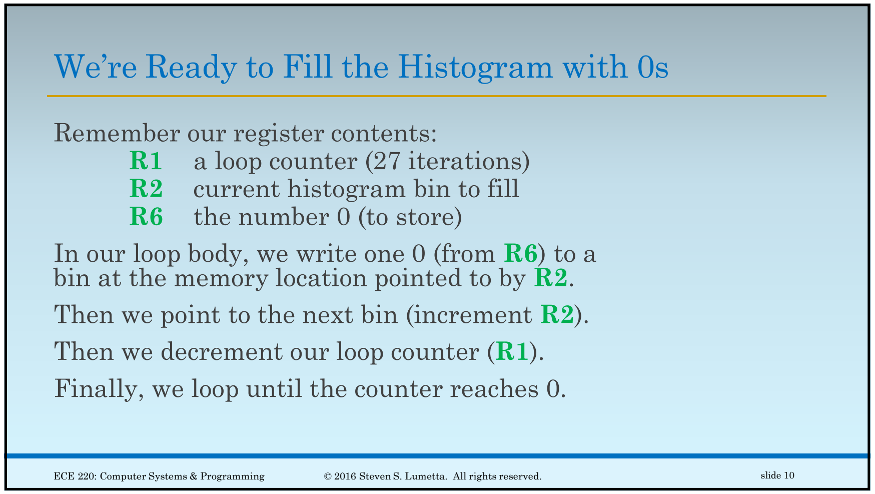
Prepare Our Registers to Initialize the Histogram

```
.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:

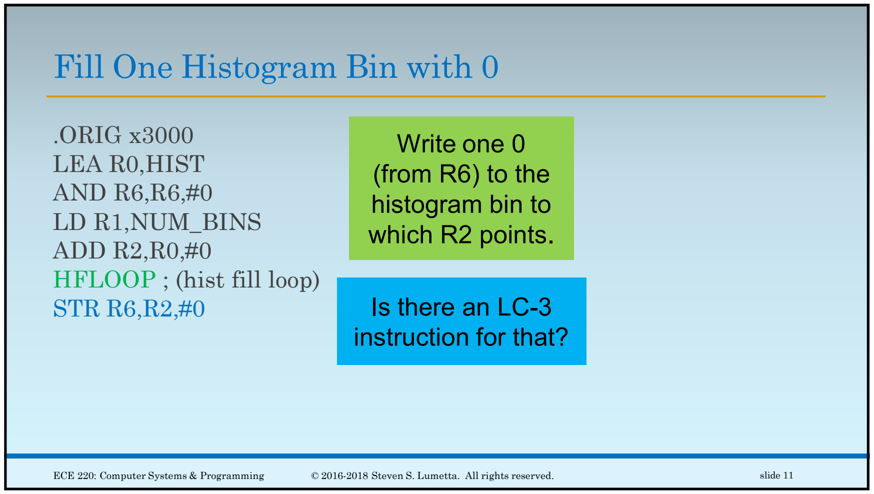
R1	a loop counter (27 iterations)
R2	current histogram bin to fill
R6	the number 0 (to store)

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.

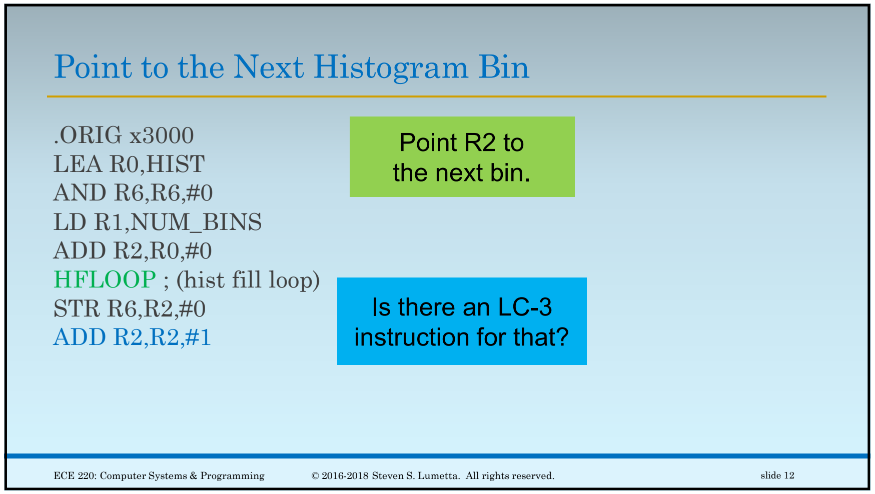


Fill One Histogram Bin with 0

```
.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
```

Write one 0 (from R6) to the histogram bin to which R2 points.

Is there an LC-3 instruction for that?

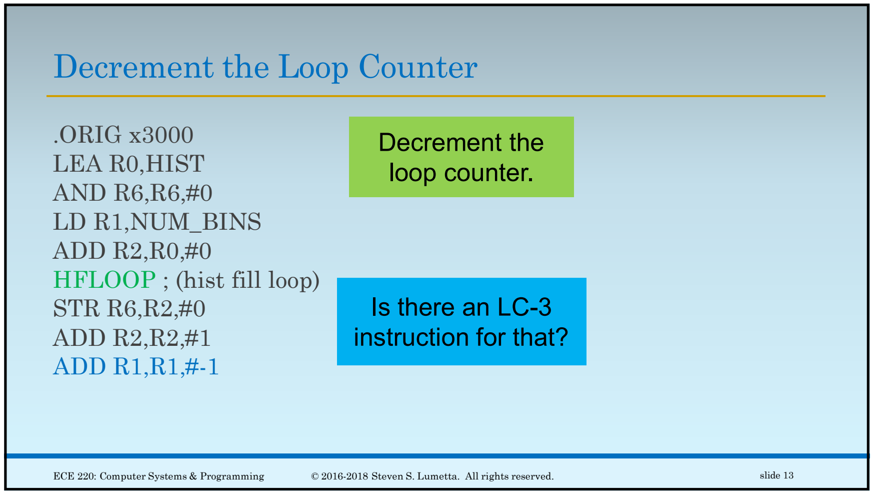


Point to the Next Histogram Bin

```
.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
```

Point R2 to the next bin.

Is there an LC-3 instruction for that?

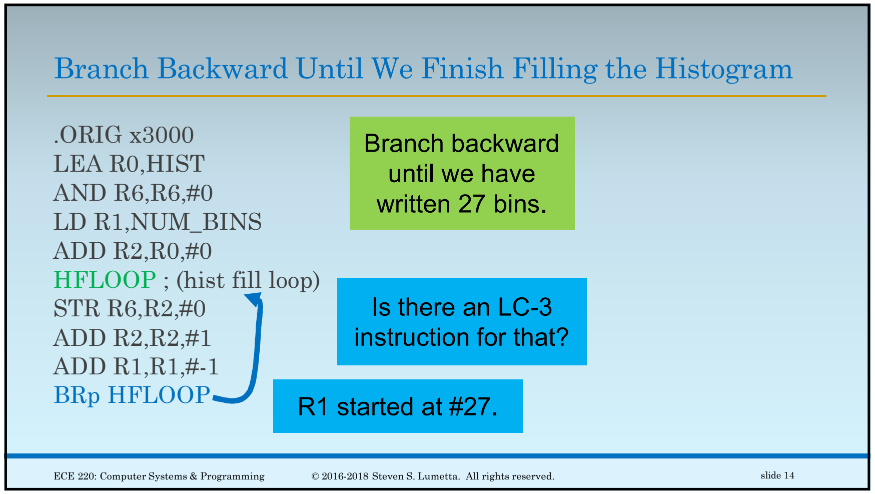


Decrement the Loop Counter

```
.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
ADD R1,R1,#-1
```

Decrement the loop counter.

Is there an LC-3 instruction for that?



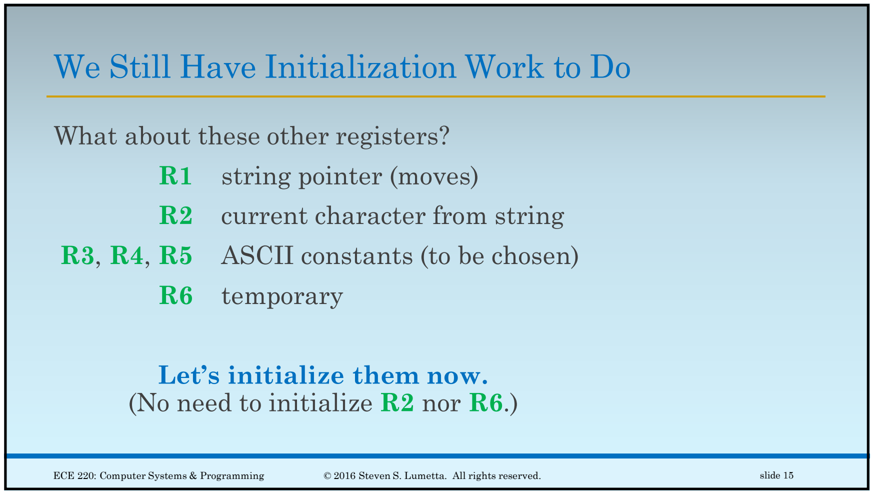
Branch Backward Until We Finish Filling the Histogram

```
.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
ADD R1,R1,#-1
BRp HFLOOP
```

Branch backward until we have written 27 bins.

Is there an LC-3 instruction for that?

R1 started at #27.

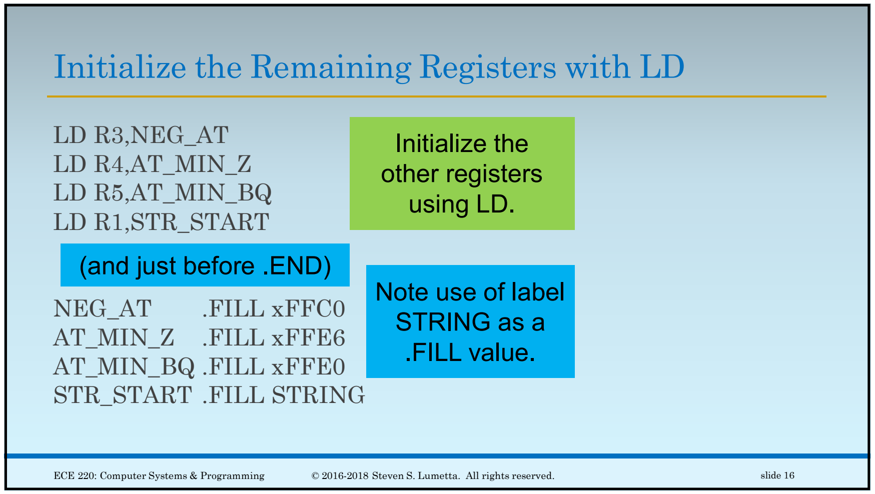


We Still Have Initialization Work to Do

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

```
LD R3,NEG_AT
LD R4,AT_MIN_Z
LD R5,AT_MIN_BQ
LD R1,STR_START
```

Initialize the other registers using LD.

(and just before .END)

```
NEG_AT      .FILL xFFC0
AT_MIN_Z    .FILL xFFE6
AT_MIN_BQ   .FILL xFFE0
STR_START   .FILL STRING
```

Note use of label STRING as a .FILL value.

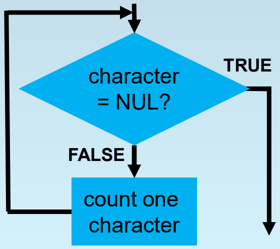
Ready to Count Letters?

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
```

Load a character from the string.

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

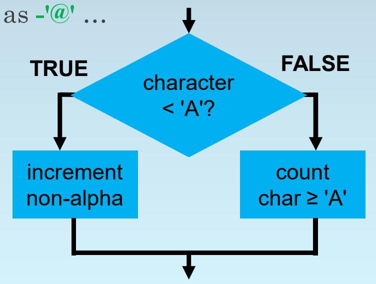
```
COUNTLOOP
LDR R2,R1,#0
BRz DONE
```

Check for NUL (x00).

Now We Can Classify the Character

We need to compare with capital **A**.

Let's define **R3** as **-'@'** …



Subtract @ to Compare with Capital A

Remember the ASCII table?

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

Subtracting **'@'** allows us to check for non-alphabetic characters in the left region.

We store the difference (original character minus **'@'**) back in **R2**, so A through Z become 1 through 26.

Subtract @ to Compare with Capital A

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

Compare with capital A.

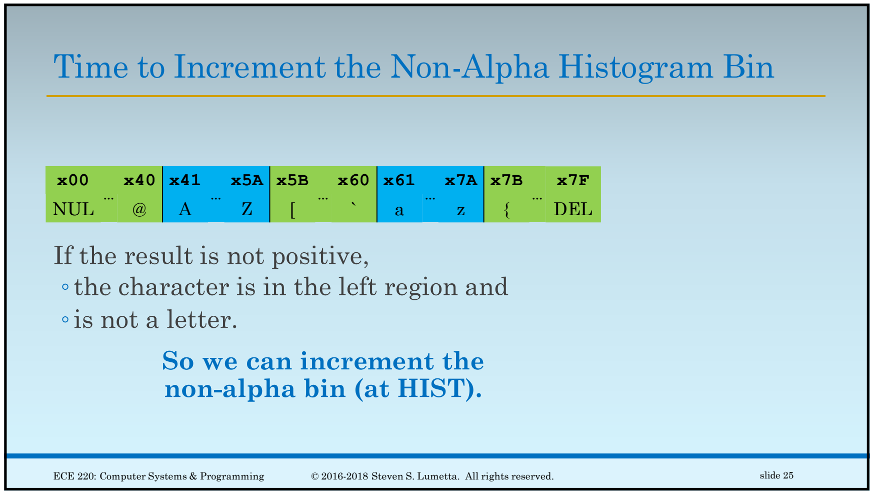
Add R3 (-'@') to R2 and write the sum back into R2.

Branch Unless We Have a Character in the Left Region

```
COUNTLOOP
LDR R2,R1,#0
BRz DONE
ADD R2,R2,R3
BRp AT_LEAST_A
```

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

What is the branch condition?



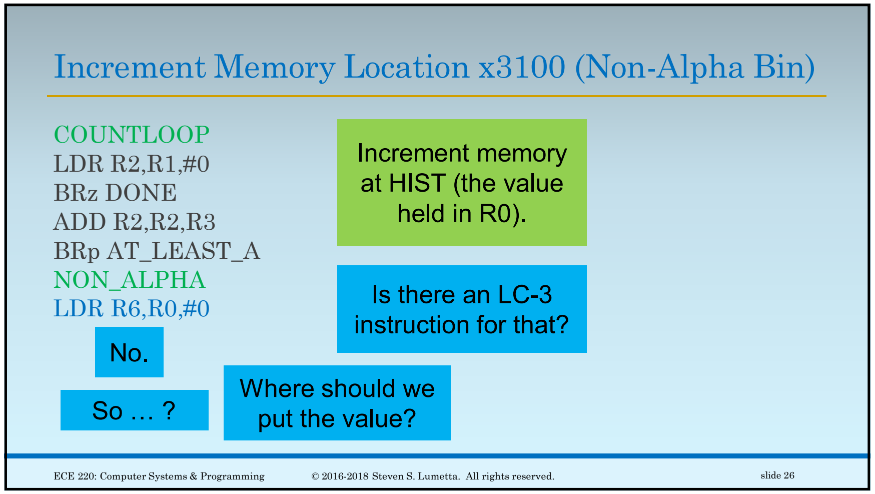
Time to Increment the Non-Alpha Histogram Bin

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

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
```

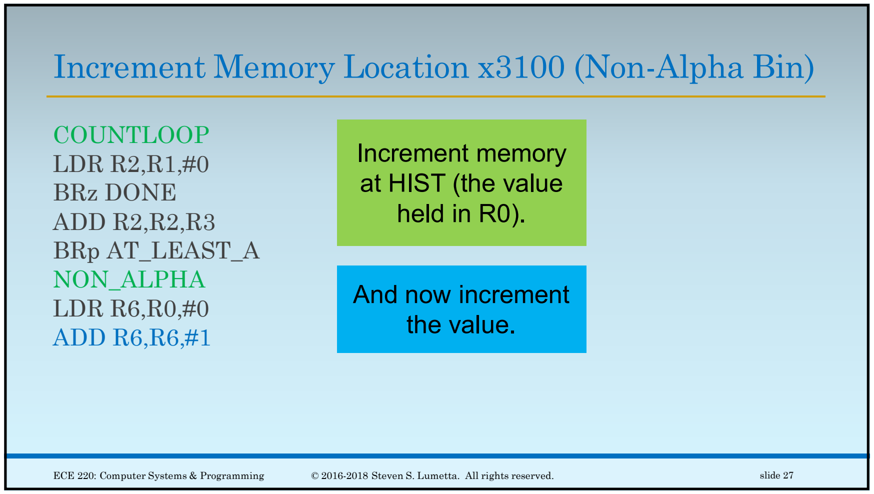
Increment memory at HIST (the value held in R0).

Is there an LC-3 instruction for that?

No.

So … ?

Where should we put the value?

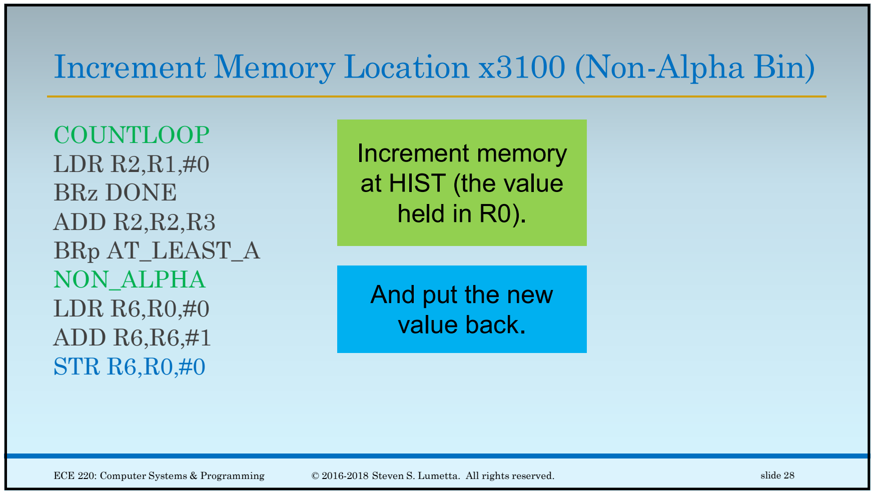


Increment Memory Location x3100 (Non-Alpha Bin)

```
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 at HIST (the value held in R0).

And now increment the value.



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 at HIST (the value held in R0).

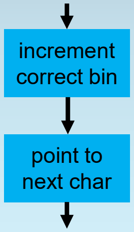
And put the new value back.

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**…



Go to the End of the Loop

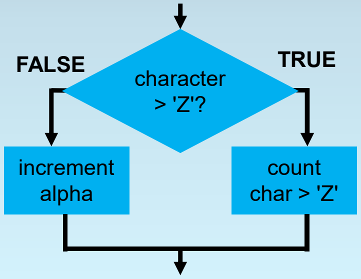
```
COUNTLOOP
LDR R2,R1,#0
BRz DONE
ADD R2,R2,R3
BRp AT_LEAST_A
NON_ALPHA
LDR R6,R0,#0
ADD R6,R6,#1
STR R6,R0,#0
BRnzp GET_NEXT
```

We are done counting this character.

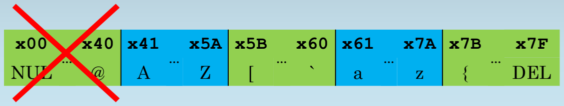
Branch (always) to the end of our loop (make up a name for it!).

We Need to Check for a Capital Letter

Next, we compare with capital **Z**.



Subtract Z to Make the Next Comparison

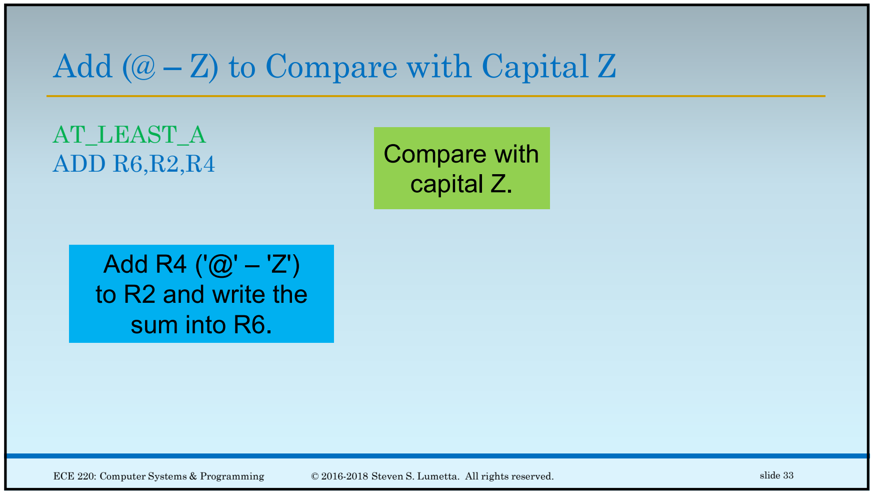


~~x00 … x40~~	x41 … x5A	x5B … x60	x61 … x7A	x7B … x7F
~~NUL … @~~	A … Z	[… `	a … z	{ … DEL

This time, we want to subtract **'Z'**.

But we already subtracted **'@'**, so now we add **'@' - 'Z'** (let's keep this value in **R4**).

We discard the result (store the result in **R6**).

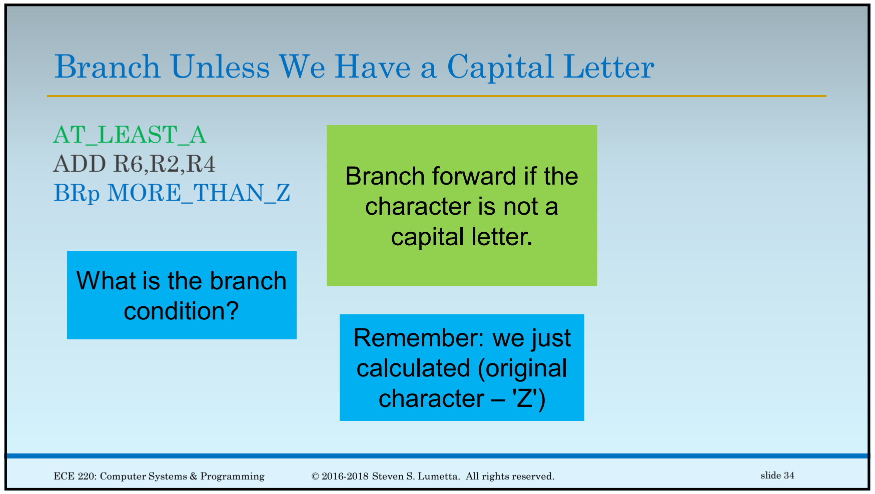


Add (@ – Z) to Compare with Capital Z

```
AT_LEAST_A
ADD R6,R2,R4
```

Compare with capital Z.

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



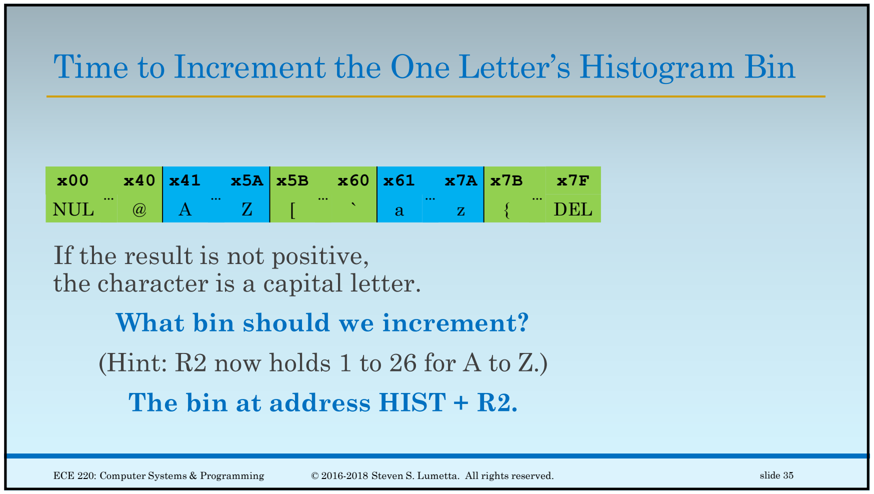
Branch Unless We Have a Capital Letter

```
AT_LEAST_A
ADD R6,R2,R4
BRp MORE_THAN_Z
```

Branch forward if the character is not a capital letter.

What is the branch condition?

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



Time to Increment the One Letter's Histogram Bin

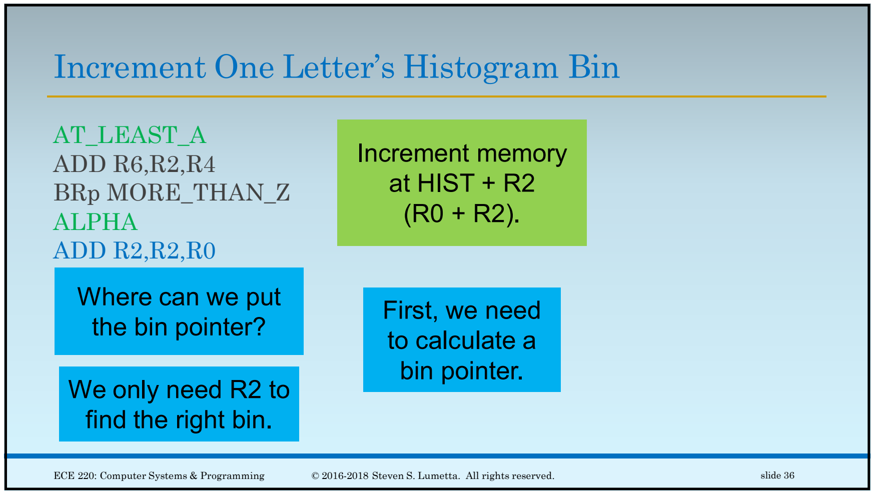
x00		x40	x41		x5A	x5B		x60	x61		x7A	x7B		x7F
NUL	…	@	A	…	Z	[	…	`	a	…	z	{	…	DEL

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.



Increment One Letter's Histogram Bin

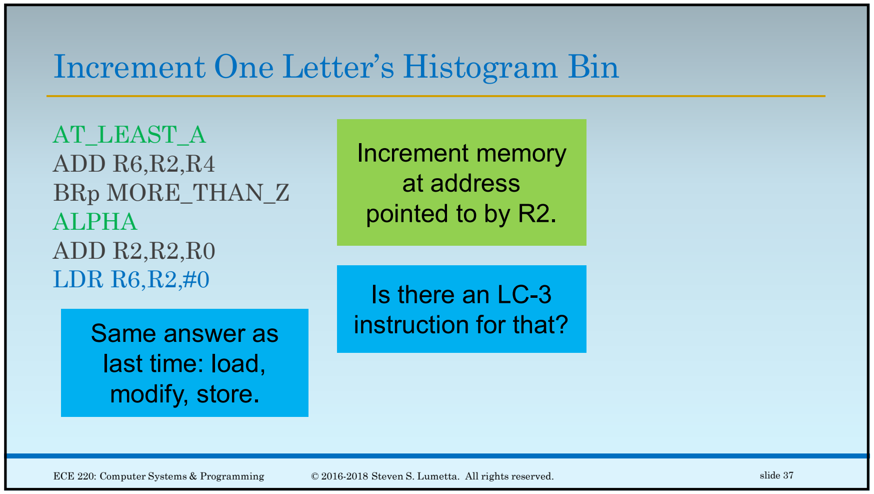
```
AT_LEAST_A
ADD R6,R2,R4
BRp MORE_THAN_Z
ALPHA
ADD R2,R2,R0
```

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

Where can we put the bin pointer?

We only need R2 to find the right bin.

First, we need to calculate a bin pointer.



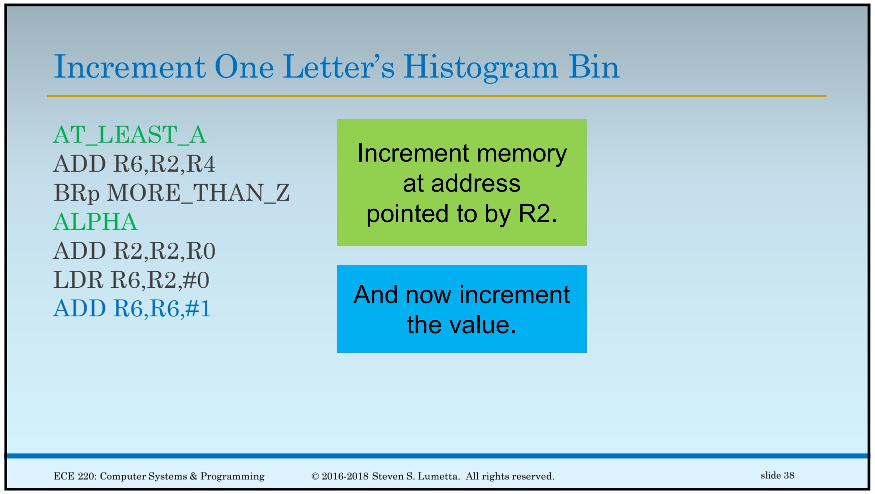
Increment One Letter’s Histogram Bin

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

Increment memory at address pointed to by R2.

Is there an LC-3 instruction for that?

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

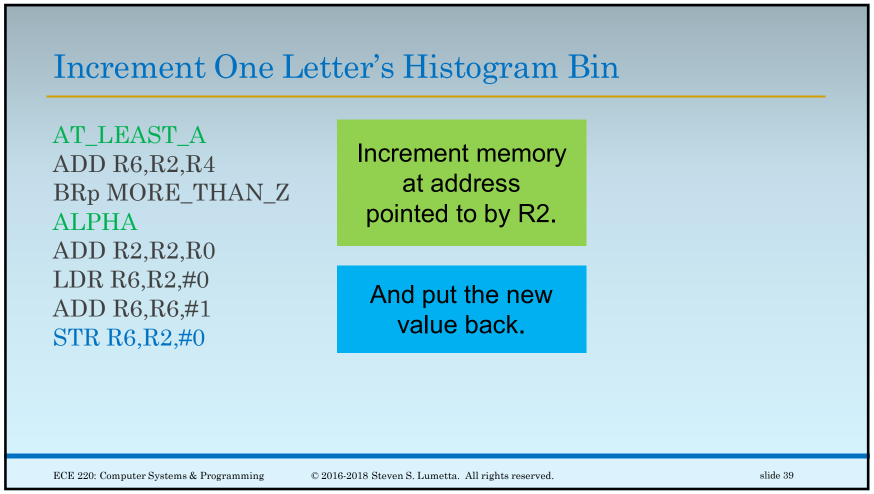


Increment One Letter’s Histogram Bin

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

Increment memory at address pointed to by R2.

And now increment the value.

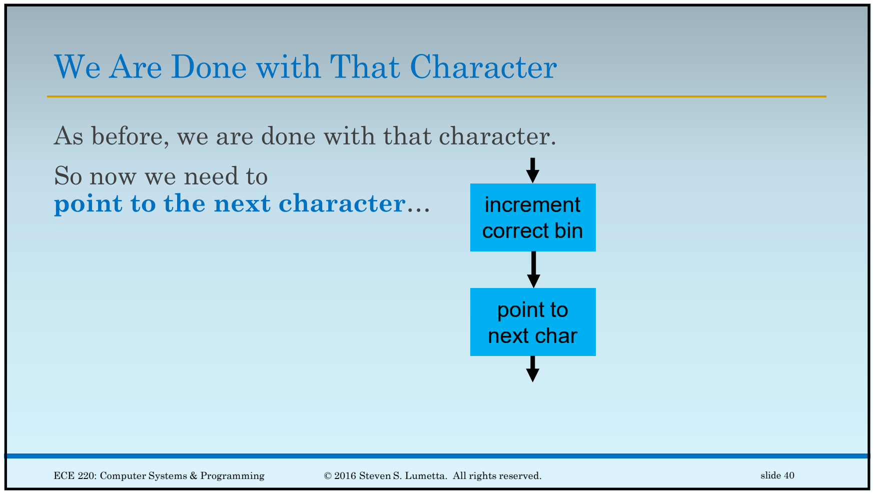


Increment One Letter’s Histogram Bin

```
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.



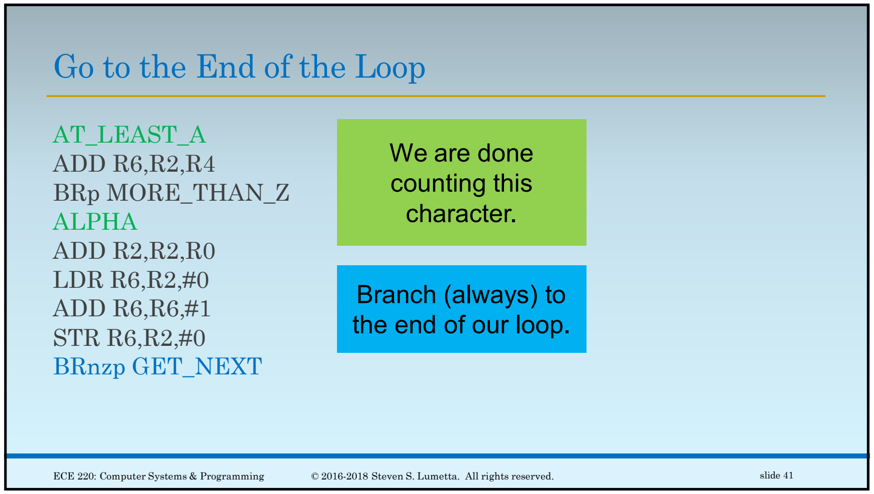
We Are Done with That Character

As before, we are done with that character.

So now we need to **point to the next character…**

increment correct bin

point to next char

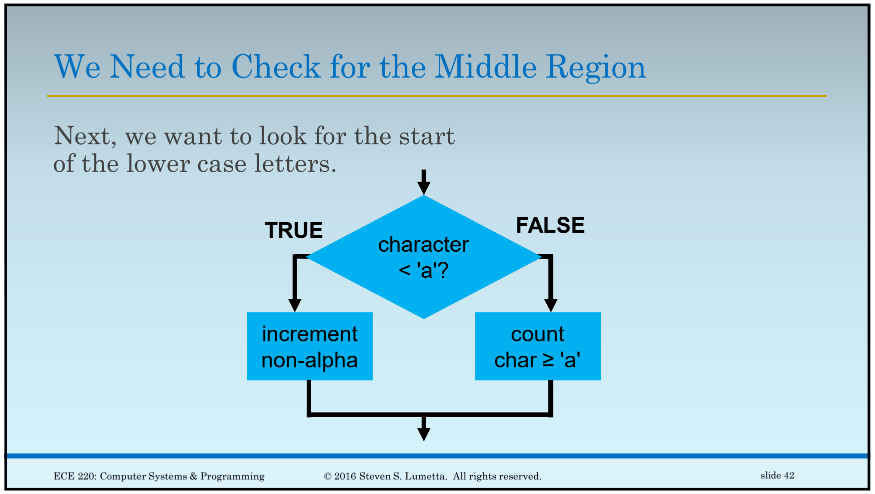


Go to the End of the Loop

```
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
BRnzp GET_NEXT
```

We are done counting this character.

Branch (always) to the end of our loop.



We Need to Check for the Middle Region

Next, we want to look for the start of the lower case letters.

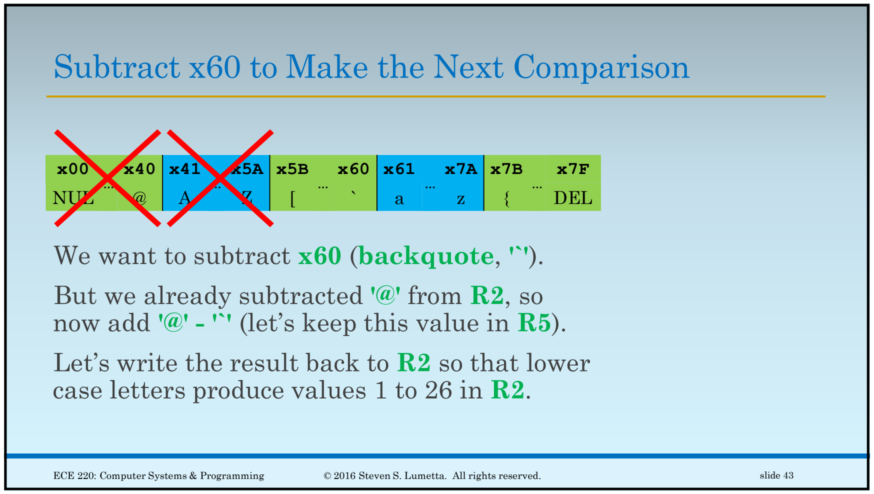
character < 'a'?

TRUE

increment non-alpha

FALSE

count char ≥ 'a'



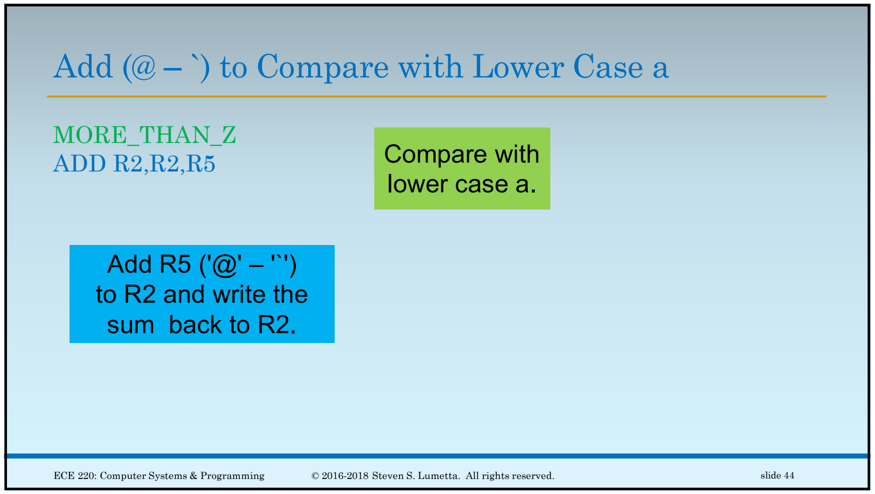
Subtract x60 to Make the Next Comparison

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

We want to subtract **x60** (**backquote**, **'`'**).

But we already subtracted **'@'** from **R2**, so now add **'@' - '`'** (let's keep this value in **R5**).

Let's write the result back to **R2** so that lower case letters produce values 1 to 26 in **R2**.



Add (@ – `) to Compare with Lower Case a

```
MORE_THAN_Z
ADD R2,R2,R5
```

Compare with lower case a.

Add R5 ('@' – '`') to R2 and write the sum back to R2.

When Do We Have a Character in the Middle Region?

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

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

How Can We Increment the Non-Alpha Bin?

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

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!

Branch If We Have a Character in the Middle Region

```
MORE_THAN_Z
ADD R2,R2,R5
BRnz NON_ALPHA
```

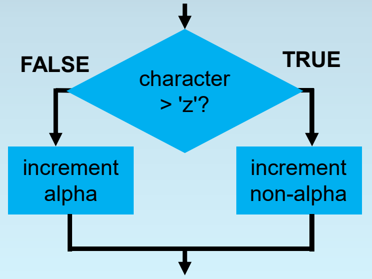
Handle characters in the middle region.

So what is the branch condition?

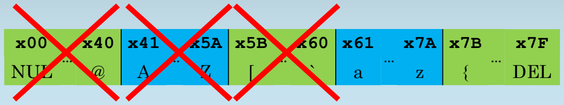
Remember the label that we created earlier (for incrementing the non-alpha bin)?

We Need to Check for a Lower Case Letter

Next, we compare with lower case **z**.



Subtract z to Make the Next Comparison



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

This time, we want to subtract **'z'**.

But we already subtracted **'`'**, so now we add **'`' - 'z'** (it's already in **R4**!).

We discard the result (store the result in **R6**).

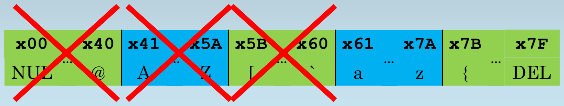
Add (` – z) to Compare with Lower Case z

```
MORE_THAN_Z
ADD R2,R2,R5
BRnz NON_ALPHA
ADD R6,R2,R4
```

Add R4 ('`' – 'z') to R2 and write the sum into R6.

Compare with lower case z.

When Do We Have a Lower Case Letter?



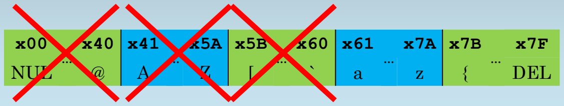
x00 … x40	x41 … x5A	x5B … x60	x61 … x7A	x7B … x7F
NUL … @	A … Z	[… `	a … z	{ … DEL

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?



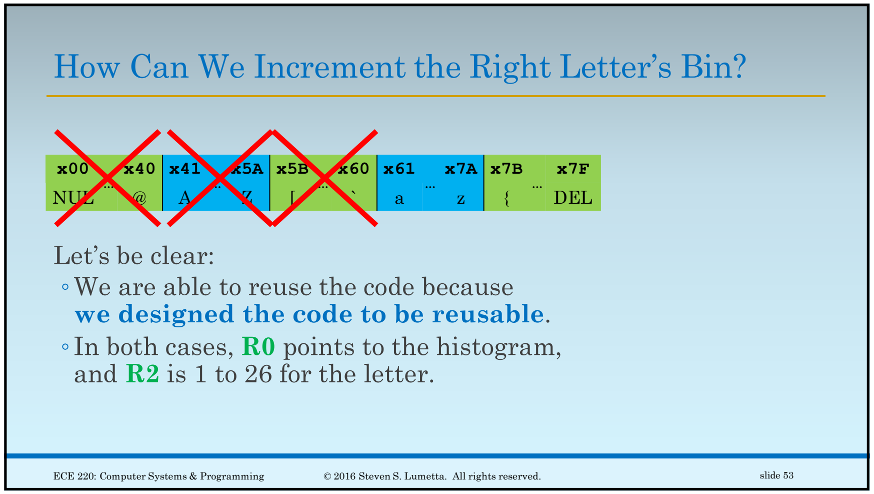
x00 … x40	x41 … x5A	x5B … x60	x61 … x7A	x7B … x7F
NUL … @	A … Z	[… `	a … z	{ … DEL

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!

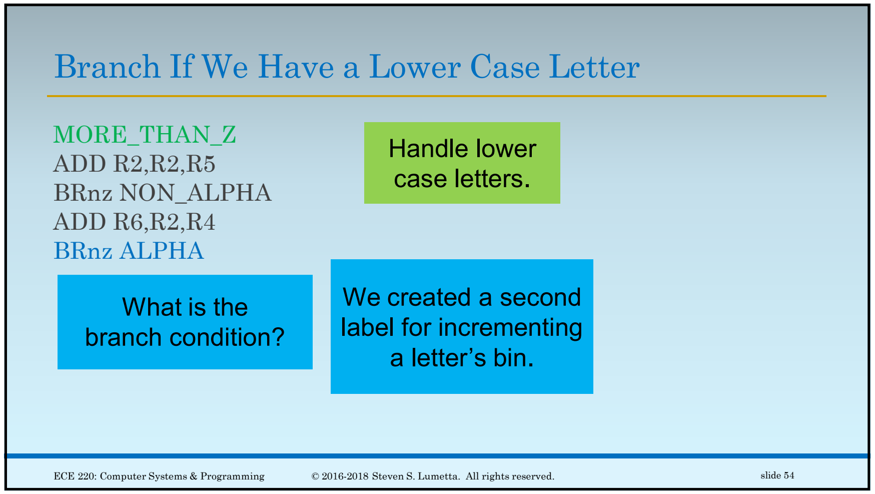


How Can We Increment the Right Letter's Bin?

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

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.



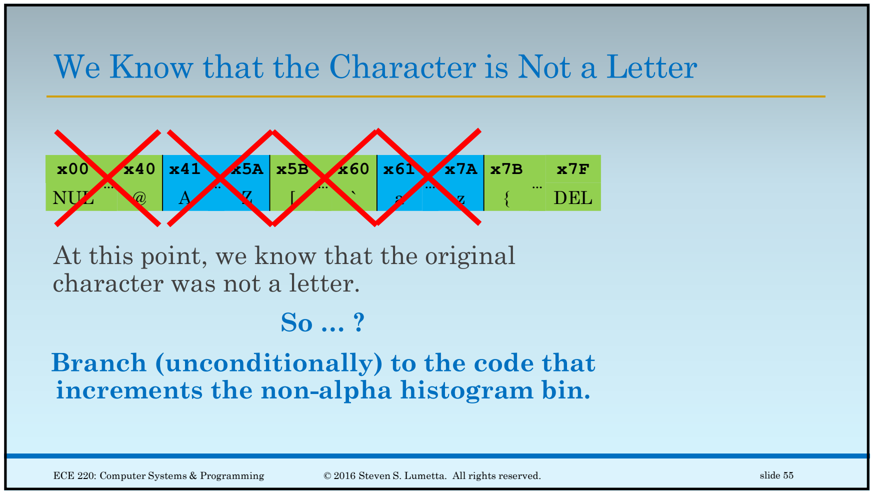
Branch If We Have a Lower Case Letter

```
MORE_THAN_Z
ADD R2,R2,R5
BRnz NON_ALPHA
ADD R6,R2,R4
BRnz ALPHA
```

Handle lower case letters.

What is the branch condition?

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



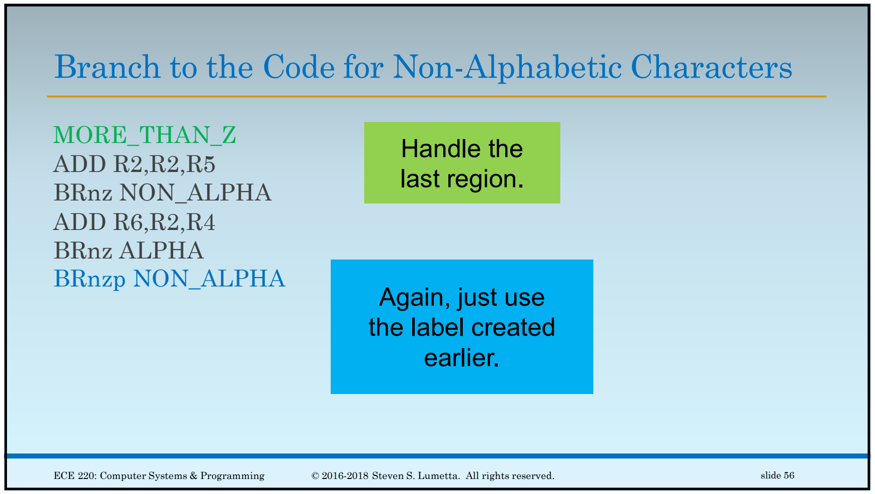
We Know that the Character is Not a Letter

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

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 to the Code for Non-Alphabetic Characters

```
MORE_THAN_Z
ADD R2,R2,R5
BRnz NON_ALPHA
ADD R6,R2,R4
BRnz ALPHA
BRnzp NON_ALPHA
```

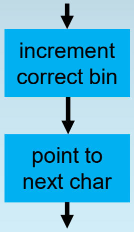
Handle the last region.

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.



Advance the String Pointer to the Next Character

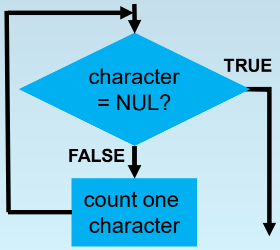
```
GET_NEXT
ADD R1,R1,#1
```

Advance the string pointer (in R1).

Is there an LC-3 instruction for that?

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.

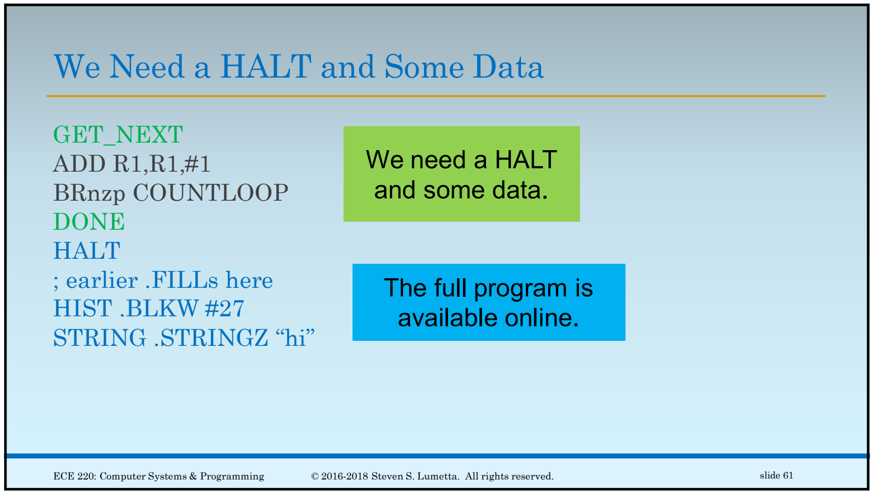


Return to the Start of the Loop

```
GET_NEXT
ADD R1,R1,#1
BRnzp COUNTLOOP
```

Return to the start of the loop.

Is there an LC-3 instruction for that?

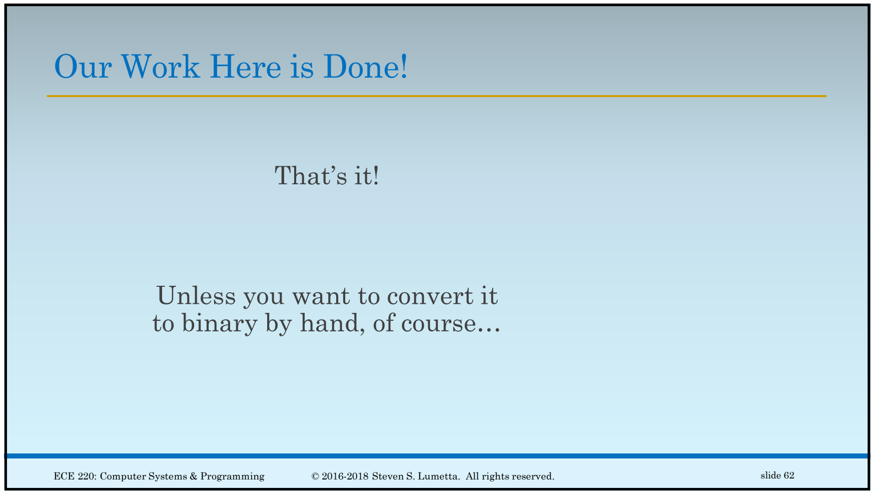


We Need a HALT and Some Data

```
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…