Want to Reuse Code, Not Rewrite Code

Problem when writing software:
- need to solve a task more than once,
- but want to write the code exactly once!

LC-3 examples:
- multiplication, division, exponentiation
- reading numbers or strings from the keyboard
- printing decimal or hexadecimal numbers to the display

An Example with Many Opportunities

A more detailed example:
- Human types in a value \( N \) followed by integral coefficients of \( F(x) \), a degree \( N \) polynomial in \( x \).
- Starting near \( x=0 \), and going in both directions until \( F(x) \) overflows a 16-bit 2's complement representation, find the integral values of \( x \) surrounding any roots in the polynomial in the specified region.

Now: how many times did you want to write multiplication?

Letter Frequency Example Reused, but Converged

The letter frequency example reused code to increment bins:
- one block of code for the non-alpha bin (used for three separate regions of ASCII),
- and one block of code for the alpha bins (used for both upper- and lower-case letters).

But in all cases, control flow converged to GET_NEXT.
How Can We Change PC to Different Values?

What if control flow does not converge?

; read degree N from keyboard
; loop from 0 to N
; read coefficient from keyboard
; continue with next loop iteration

After reading a number, does PC change to start the loop, or to continue the loop?

Use the JMP Instruction to Go to the Right Place

Remember the JMP instruction?

\[
\text{JMP BaseR} \quad ; \quad \text{PC} \leftarrow \text{BaseR}
\]

Use a register—R7, for example—

* to point to the next instruction
* after executing our common code.

Then,

* at the end of the common code,
* execute JMP R7.

Two Instructions Suffice to Execute the Common Code

Say that the code for reading a number starts at READNUM.

; read degree N from keyboard
LEA R7,#1
BRnzp READNUM
; loop from 0 to N
; read coefficient from keyboard
; continue with next loop iteration

Can Use the Same Sequence Many Times

Write the same two instructions for the other use of READNUM.

; read degree N from keyboard
LEA R7,#1
BRnzp READNUM
; loop from 0 to N
; read coefficient from keyboard
LEA R7,#1
BRnzp READNUM
; continue with next loop iteration
Most ISAs Have Special Instructions for Subroutines

The `READNUM` code in the example is called a subroutine.

The notion of subroutines:
- also called procedures, functions, methods, and so on
- is so important that
- most ISAs have special instructions to support subroutines.

LC-3 also has such instructions.

JSR and JSRR Call Subroutines in LC-3

LC-3 offers two ways to call a subroutine: JSR (top form), and JSRR (bottom form).

Notice that either can replace our two-instruction sequence, and both change R7.

```
0 1 0 0 1
PCoffset11

R7 ← PC, PC ← PC + SEXT16(PCoffset11)
```

Call Subroutines with JSR(R), Return with RET

Using JSR, we now have...

```plaintext
; read degree N from keyboard
JSR READNUM
; loop from 0 to N
; read coefficient from keyboard
JSR READNUM
; continue with next loop iteration
Return from subroutine is still JMP R7,
But the assembler allows pseudo-op RET.
```