Humans vs. Computers

The Church-Turing Hypothesis tells us:
◦ anything a human can compute
◦ can also be computed by a computer
◦ (and vice-versa).
So…

What’s the difference between humans and computers?

The Answer

Humans are smart.
Computers are dumb.

What’s this?

Some Problems are Hard to Solve Systematically

Computers have difficulty solving that kind of problem.
Or, rather, programmers have difficulty
◦ knowing how their brains solve such problems,
◦ so they can’t get a computer to solve such problems quickly.
That’s why we can use such problems to check for human presence.
Beware of Anthropomorphism

I may have said (and may still say) sentences like...

“The LC-3 only understands 2’s complement.”

But the LC-3 is not human.
The LC-3 “understands” nothing.

So what am I trying to say?

LC-3 Includes Operations on 2’s Complement Values

“The LC-3 only understands 2’s complement.”

By the definition of the LC-3 ISA, many constants and values are treated as 2’s complement.

Any LC-3 microarchitecture needs hardware designed to support 2’s complement.

For example, notice the numerous sign extension boxes in Patt and Patel’s datapath.

Other Data Types Must Be Handled in Software

“The LC-3 only understands 2’s complement.”

In contrast, there are no instructions (nor hardware) for directly manipulating bits in other representations.

How do we use other data types with an LC-3 processor?

Translate operations on other data types into sequences of instructions.

In other words, write software to do it.

Another Example: Adding Strings

Here’s a software representation for a string of text (the string is “19”).

The address of the first ASCII character in memory, x4012, is used to represent the string.

To “read” the string,
  ° look at consecutive memory locations
  ° until we find a 0 (an ASCII NUL character),
  ° which indicates the end of the string.
Can We Add Two Strings?

Here's another string. What is it? “23”

Say that the LC-3 executes:

\[
\begin{align*}
R1 & \leftarrow x4012 \\
R2 & \leftarrow x7196 \\
R3 & \leftarrow R1 + R2
\end{align*}
\]

What is R3? xB1A8

What is stored at xB1A8? Bits!

You Understand Why Adding Addresses Doesn’t “Work”

Obviously, if we want to add two strings that represent numbers, we need to do more work.

Unfortunately, many people:
- who have never seen representations
- nor how computers work
- make this kind of mistake
- and struggle to understand why the answer is not what they expect.

Lend Me Your Brains for a Minute?

I almost forgot!

I need to ask your help again!

Can you help me sort these numbers?

“41,962” “41321” “9874”

biggest  middle  smallest

Are You Sure About Your Answers?

Hmm. Are you sure?

I just ask because, well …

I asked my computer, too.

And it gave different answers:

“41,962” “41321” “9874”

humans  biggest  middle  smallest
computers  smallest  middle  biggest
A Side-by-Side Comparison of the Numbers

Let’s compare them side by side.

41,962
41321
9874

What’s bigger, “4” or “9?”

Oh, so “9874” is the biggest!
Please be more careful when you help me!

What’s the next largest?

41,962
41321
9874

“4” is equal to “4.”
“1” is equal to “1.”

What’s bigger, “,“ or “3?”

Ah, so “41321” is the middle value. Good.

So the Computer is Right?

It seems that the computer is right.
At least, for some definition of “right.”
This type of answer is what you get if you sort strings in ASCII order (instead of alphabetical order).

“41,962” “41321” “9874”

humans biggest middle smallest
computers smallest middle biggest

Remember: Computers are Dumb

Think it’s just a silly example?

Take a look at the index of Patt and Patel.

Computers do exactly what they are told.