Table of Squares Example

Here, we are going to take a look at a specific example of how to use the ENIAC simulator to compute a table of squares and to punch the output onto cards. We're going to follow the typical development cycle of first analyzing the problem to create an algorithm to solve it. Then we'll create an ENIAC configuration that implemented the algorithm, and finally translate that configuration into a input file for the simulator. The resulting simulator configuration can be found in the `programs` directory in the file named `sq3.e`.

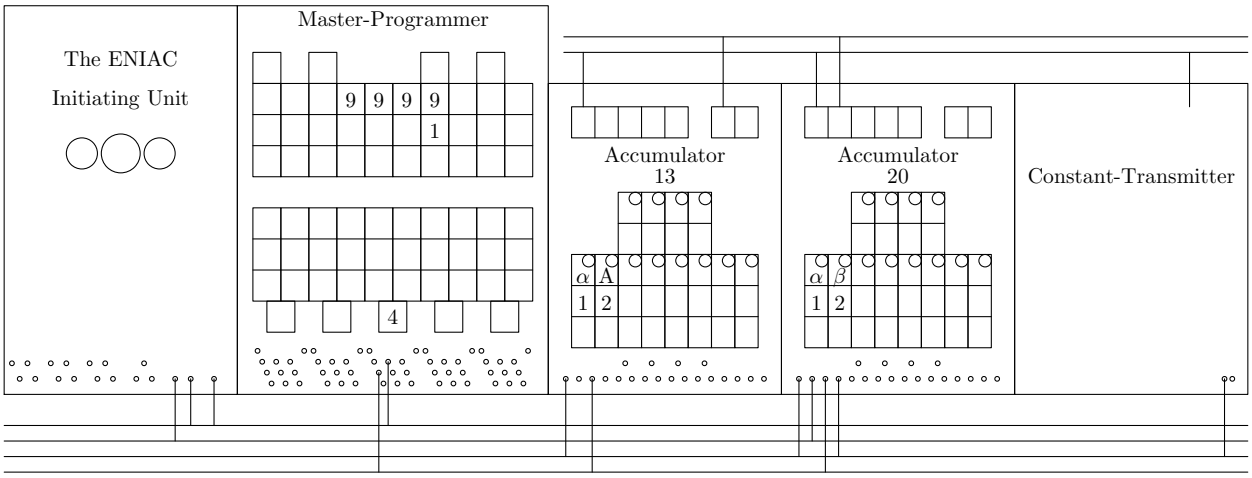
Problem Analysis

Since the task is to compute a table of $f(x) = x^2$ values for integer values of x, the first thought might be to use the multiplier to multiply each x value by itself. However, there's a faster way of doing this by taking advantage of the familiar equation: $(x+1)^2 = x^2 + 2x + 1$. In other words, if we have the values of $f(x) = x^2$ and x, then all we have to do is add $2x + 1$ to $f(x)$ to get $f(x+1)$. So in algorithmic terms, we might describe the process as follows:

Initialize the values $f(x) = 0$ and $x = 0$
For each row of the table:
  Add $2x$ to $f(x)$
  Add 1 to $f(x)$
  Add 1 to x
  Punch a card with the values of x and x^2

Implementing on the ENIAC

Because after clearing the machine, all the accumulators are set to 0, we'll consider our initialization step solved. For the 1 that shows up in a couple of steps of the algorithm, we'll set up the constant transmitter with the value 1 and use it for those cases where we need to add 1. The next decision we need to make is how the programming of the master programmer is going to relate to the "for each" loop. I.e. is this going to be a pre-test or a post-test loop. For this example, we're going to choose pre-test. With these decisions made, we'll configure the ENIAC according to this figure, explained in notes that follow:



1. The data trunks are the lines above the accumulators and constant transmitter and are numbered from bottom to top starting at 1.
2. The program trunks are below all the units and are numbered from 1-1 to 1-4 top to bottom.
3. Since we're treating this as a pre-test loop, the initiating output of the initiating unit is carried through program trunk 1-1 to the stepper input (Ci) of the third (C) stepper of the master programmer. Because of the switch settings we've made, the first 9999 such input pulses will result in output pulses on the C1o output.
4. The C1o output goes to program trunk 1-4 that starts the sequence of operations in the body of the loop.

5. The first step is to add $2x$ to x^2, were x is in Accumulator 13 and x^2 is in Accumulator 20. Program trunk 1-4 then starts program 6 on both Accumulators 13 and 20. Accumulator 13 responds by transmitting its value on the A output (to data trunk 2) twice while Accumulator 20 receives from data trunk 2 into its β input twice. As a result, we now have $x^2 + 2x$ in Accumulator 20.
6. The completion of this operation is signaled on Accumulator 20's 6o line which is connected to program trunk 1-3.
7. The signal on program trunk 1-3 initiates program 5 on both Accumulators 13 and 20 as well as initiating program 26 on the constant transmitter. The behavior of these programs is to put the number 1 onto data trunk 1 and to read the value on data trunk 1 into both accumulators via their α inputs. As a result, we now have $x + 1$ in Accumulator 13 and $x^2 + 2x + 1 = (x + 1)^2$ in Accumulator 20.
8. With the completion of these calculations, we can now puch an output card with the values of x and x^2 on it. The completion of the adds of 1 results in a pulse on the 5o line of Accumulator 20 which is carried to the Pi input of the initiating unit via program trunk 1-2.
9. The Pi input on the initiating unit starts a card punch cycle. If the card punch is idle, then the data on the accumultors is read into the punch interface and a pulse is sent out the Po terminal to indicate that the next computation can begin. If the card punch is still busy with the last card, the Po pulse is delayed until the card punch has completed enough of its operation to allow the accumulator values to again be copied into the punch interface.
10. The Po output from the initiating unit is connected to program trunk 1-1 in the same way as the initiating output. As a result, when the punch control indicates it's time to start a new computation, the signal on program trunk 1-1 starts the whole process over, just like the original initiation signal did.

Simulator Configuration

The first few lines of our configuration file describe the connections on data trunk 1 from the constant transmitter output to the α inputs of the two accumulators. Note that in the file itself, you can specify the input terminal name with the UTF-8 codepoint for α, the lower-case letter 'a' or the word 'alpha.'

```
p c.o 1
p 1 a13.α
p 1 a20.α
```

The next two lines are the connection from the A output of Accumulator 13 to the β input of Accumulator 20.

```
p a13.A 2
p 2 a20.β
```

Now we move on to the program control connections. The first two connect the initiating pulse output to the stepper input on the master programmer.

```
p i.io 1-1
p 1-1 p.Ci
```

The completion of the constant transmitter program triggers the rest of the computations. This is carried on program trunk 1-4. As described above, the control then sequences through Program 6 of Accumulators 13 and 20, to Program 5 of both accumulators to the punch.

```
p p.C1o 1-4
p 1-4 a13.6i
p 1-4 a20.6i
p a20.5o 1-2
p a20.6o 1-3
p 1-3 a13.5i
p 1-3 a20.5i
p 1-3 c.26i
p 1-2 i.pi
p i.po 1-1
```

Now we set up the switches, the first set of which are the operation and repeat switches for Programs 5 and 6 on Accumulators 13 and 20.

```
s a13.op5 α
s a13.op6 A
s a13.rp5 1
s a13.rp6 2
s a20.op5 α
s a20.op6 β
s a20.rp5 1
s a20.rp6 2
```

The last switch settings for the constant transmitter are not shown on the diagram given here. They are set so that Program 26 transmits the value of the J constant, and so that the J constant is set to the value 1.

```
s c.s26 Jlr
s c.j1 1
```

The next several switch settings are those for the master programmer as shown in the diagram.

```
s p.d17s1 9
s p.d16s1 9
s p.d15s1 9
s p.d14s1 9
s p.d14s2 1
s p.cC 4
```

Finally, we have the switch settings on the punch interface that enable the printing of the values of Accumulators 13 and 20.

```
s pr.2 P
s pr.3 P
s pr.15 P
s pr.16 P
```